NPS ARCHIVE
1997.09
HALLBERG, E.

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DISSERTATION

ON INTEGRATED PLANT, CONTROL AND GUIDANCE DESIGN

by

Eric N. Hallberg

September, 1997

Thesis Advisor: Isaac I. Kaminer

Thesis
H1612515

Approved for public release; distribution is unlimited.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, Va 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September, 1997	Ph.D. Dissertation

4. TITLE AND SUBTITLE ON INTEGRATED PLANT, CONTROL AND GUIDANCE DESIGN

5. FUNDING NUMBERS

6. AUTHORS Hallberg, Eric, N.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT(*maximum 200 words*)

Two theoretical methods and the development of a guidance, navigation and control rapid protoyping system address the issue of considering the integral participation of feedback early in the design process. The first method addresses the problem of sizing the horizontal tail on a statically unstable transport aircraft. Dynamic constraints including recovery from a severe angle of attack excursion and penetration of a vertical wind shear are formulated in terms of the solution to a convex minimization problem utilizing LMIs and used to size the horizontal control surfaces. The second method addresses the problem of tracking inertial trajectories with applications for unmanned air vehicles. This problem is posed and solved within the framework of gain scheduled control theory leading to a new technique for integrated guidance and control systems with guaranteed performance and robustness properties. Finally, a rapid prototyping system for the flight test of GNC algorithms for unmanned air vehicles is designed that affords a small team the ability to quickly take a new concept in guidance, navigation and control from initial conception to flight test.

14. SUBJECT TERMS Plant Controller Optimization, Horizontal Tail Sizing, LMIs, Integrated Guidance and Control, Inertial Trajectory Tracking, Gain Scheduled Control, Parameter Identification, Rapid Prototyping Convex Optimization, Robust Control

15. NUMBER OF PAGES 210

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

ON INTEGRATED PLANT, CONTROL AND GUIDANCE DESIGN

by

Eric N. Hallberg
Lieutenant Commander, United States Navy
B.S., University of Pennsylvania, 1984
M.S., Naval Postgraduate School, 1994

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN AERONAUTICS AND ASTRONAUTICS

from the

NAVAL POSTGRADUATE SCHOOL

September, 1997

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

ABSTRACT

Two theoretical methods and the development of a guidance, navigation, and control rapid protoyping system address the issue of considering the integral participation of feedback early in the design process. The first method addresses the problem of sizing the horizontal tail on a statically unstable transport aircraft. Dynamic constraints, which implicitly involve the flight control system, may prove more restrictive than traditional static constraints. Recovery from a severe angle of attack excursion, or penetration of a vertical wind-shear, are formulated in terms of the solution to a convex minimization problem utilizing LMIs, and numerically solved. The second method addresses the problem of tracking inertial trajectories, with applications for unmanned air vehicles. This problem is posed and solved within the framework of gain-scheduled control theory. This leads to a new technique for integrated guidance and control systems, with guaranteed performance and robustness properties. Finally, a rapid prototyping system for the flight testing of guidance, navigation, and control algorithms for unmanned air vehicles is designed. The system affords a small team the ability to take a new concept in guidance, navigation, and control from initial conception to flight test. A proof-of-concept demonstration of the system is detailed when the new integrated guidance and control algorithm previously described is tested in flight on an unmanned air vehicle.

TABLE OF CONTENTS

I. OVERVIEW 1

A. LAYOUT OF THE CHAPTERS 1

B. CONTRIBUTION 2

II. INTEGRATED PLANT CONTROLLER OPTIMIZATION . . 3

A. INTRODUCTION 3

1. Example of a Design Space 5

B. BACKGROUND IN LINEAR MATRIX INEQUALITIES 8

1. State Feedback Synthesis LMIs 12

2. Output Feedback Synthesis LMIs 14

C. HIGH ANGLE OF ATTACK RECOVERY 20

1. Problem Formulation for a Rigid Body HSCT 20

2. Canard Configuration 24

3. Proposed Numerical Solution 26

4. Results - Rigid Body HSCT 30

5. Problem Formulation - Aeroelastic Model 37

6. Results - Aeroelastic Model 48

D. GUST RECOVERY 52

1. Additional LMI Considerations 53

2. Problem Formulation 54

3. Proposed Numerical Solution 57

4. Gust Recovery Results 58

E. CONCLUSIONS 60

III. INTEGRATED GUIDANCE AND CONTROL 63

A. INTRODUCTION 63

B. RIGID BODY DYNAMICS 67

C. GENERALIZED ERROR DYNAMICS 72

1. Trimming Trajectories 72
2. Trimming Trajectory Coordinate System 74
3. Generalized Error Vector 77
D. TRAJECTORY TRACKING CONTROL SYSTEM DESIGN AND IMPLEMENTATION 84
1. Linear Controller Design 84
2. Gain Scheduled Controller Design 86
3. Implementation Procedure 93
E. EXAMPLE 95
1. Design Requirements and Linear Controller Design . . . 96
2. Implementation and Simulation Results 98
F. CONCLUSIONS 99
IV. RAPID PROTOTYPING SYSTEM FOR FLIGHT TEST OF AN UAV 101
A. INTRODUCTION 101
B. SYSTEM DESCRIPTION 102
1. RFTPS Capabilities 102
2. Cost, Safety and Other Considerations 103
3. Components 105
C. TRAJECTORY CONTROL: AN APPLICATION 108
1. Theoretical Background for Model Identification 109
2. Vehicle Model Identification 111
3. Autopilot Model Identification 112
4. Design of the Controller 114
5. Nonlinear Implementation 120
D. CONTROLLER IMPLEMENTATION FOR FLIGHT TEST . . 123
1. Additional Implementation Issues 124
E. RESULTS AND ANALYSIS 129

1. First Flight Test 130
2. Second Flight Test 133
F. CONCLUSIONS 135
APPENDIX A. THE TAIL SIZING DESIGN TOOL 139
1. INTRODUCTION 139
2. GENERATING A DATABASE 140
3. VIEWING A DATABASE 158
4. THE DYNAMIC AIRCRAFT MODEL 165
APPENDIX B. RFTPS SUPPORT 173
1. INTRODUCTION 173
2. SERIAL MODULE 173
LIST OF REFERENCES 187
INITIAL DISTRIBUTION LIST 191

LIST OF TABLES

I. Eigenvalues of Aeroelastic and Rigid-body Models 47

II. Matrix of Aircraft Definitions Tested 61

III. Power Budget of Onboard Components 107

IV. Comparison of selected longitudinal derivatives. 111

V. Comparison of selected lateral derivatives. 113

VI. Error Comparison for Flight of June 20, 1997: Flight Test & Simulation 133

VII. Error Comparison for Flight of July 23, 1997: Flight Test & Simulation 134

LIST OF FIGURES

1. Range of Feedback Gains 6
2. Family of LTI systems with a single acceptable gain. 7
3. Example of a *Design Space* 7
4. Region $\mathcal{L}$ 11
5. Definition of $\bar{V}_H$. 21
6. Acceptable Short Period Pole Locations 23
7. Region $\mathcal{L}$ for Level II Flying Qualities 23
8. Fixed Tail Volume and Peak Actuator Rate Limit 30
9. Fixed Tail Volume, Sweep of Peak Actuator Rates 31
10. Two Tail Volumes, Sweep of Peak Actuator Rates 32
11. 3D *Tail Sizing Design Space* 33
12. 3D *Tail Sizing Design Space* with Peak Actuator Rate Limit . . 33
13. 2D Slice of the *Tail Sizing Design Space* 34
14. 3D *Tail Sizing Design Space* with Canard 35
15. *Tail Sizing Design Space* With and Without a Canard 35
16. 2D Slice of *Tail Sizing Design Space* With and Without a Canard 36
17. Decrease In Total Control Volume Through the Addition of a Canard 36
18. 3D *Tail Sizing Design Space* - Dynamic Controller 37
19. *Tail Sizing Design Space Comparison*: Static versus Dynamic Controllers 37
20. 2D Comparsion: Static versus Dynamic Controller 38
21. Pitch Rate at Cockpit, Aeroelastic vs. Rigid Body 47
22. 1^{st} Symmetric Mode Shape 49
23. 3D *Tail Sizing Design Space*: Aeroelastic Model 49
24. *Tail Sizing Design Space*: Aeroelastic versus Rigid Body Model . 50

25. *Tail Sizing Design Space* With Canard 50
26. *Tail Sizing Design Space Comparison*: Canard On and Off . . . 51
27. 2D Comparison: Canard On and Off 51
28. Decrease In Total Control Volume Through the Addition of a Canard 52
29. 2D *Tail Sizing Design Space* Comparison: Static versus Dynamic Controller on an Aeroelastic Model 53
30. Two Wind shear Profiles 55
31. Increasing wind shear peak velocity moves the center-of-gravity limit forward. The aft line corresponds to 30 fps, the middle to 45 fps and the forward line to 60 fps while the penetration distance was held constant at 1 $\bar{c}$. 59
32. Increasing wind shear penetration distance moves the center-of-gravity limit aft. The aft line corresponds to a penetration distance of $2\bar{c}$, the middle to $1\frac{1}{2}\bar{c}$ and the forward line to $1\bar{c}$ while the peak gust velocity was held constant at 45 fps. 60
33. Set of Linear Controllers 85
34. Gain Scheduled Controller 87
35. Synthesis Model 97
36. The trajectory tracked in simulation. 99
37. Time history of position errors, Euler angles and airspeed along the trajectory. 100
38. *Frog*: The unmanned air vehicle at the Naval Postgraduate School. 103
39. The base station of the RFTPS in use at the airfield. 104
40. RFTPS Hardware Architecture 105
41. An elevator doublet is used to excite the longitudinal dynamics of *Frog*. Test flight data is compared with simulation results to assess validity of the model. 112

42. An aileron doublet is used to excite the lateral dynamics of *Frog*. Test flight data is compared with simulation results to asses validity of the model. 114
43. Block diagram of the lateral channel of the inner-loop autopilot. 115
44. Flight test data is used to identify the dynamics of the autopilot. A step signal is sent to the lateral channel of the autopilot. Measured yaw rate is compared to simulation results. 115
45. Block diagram of the longitudinal channel of the inner-loop autopilot. 116
46. Bode plot of the yaw rate command to yaw rate and climb rate command to climb rate (lower graph) for *Frog* with the autopilot on. 116
47. Linear Controller Design 119
48. Root-locus for the lateral channel. 119
49. Nyquist diagram of the loop transfer function for the lateral channel. The phase margin is 112 degrees and the gain margin is 6 dB. 120
50. Nyquist diagram of the loop transfer function for the longitudinal channel. The phase margin is 145 degrees and the gain margin is infinite. 121
51. Nonlinear Implementation of the Controller 121
52. Calibrating the Uplink 127
53. Top graph: measured vehicle yaw rate (degrees per second) versus PWM signal (μ seconds) into the lateral channel of the inner loop autopilot. Bottom graph: measured vehicle climb rate (feet per minute) versus PWM signal (μseconds) into the longitudinal channel of the inner loop autopilot. 128

54. The first (near) autonomous flight of *Frog*. The commanded trajectory is shown as a solid line while the path flown is shown as a broken line. Also shown are the results from nonlinear simulation. The wind was 18 feet per second out of the west, northwest. The graph is oriented with the y-axis pointing due north. Approximately two and one half minutes of flight are shown. 131

55. Top graph: Commanded vehicle yaw rate in degrees per second from the controller along the trajectory flown on June 20, 1997. Bottom graph: Lateral error in feet as computed by the guidance algorithm along the same trajectory. 132

56. The second autonomous flight of *Frog*. Weather conditions were near ideal as *Frog* tracked the 3-D trajectory shown by the dashed line. The results from simulation are shown as a dash-dot line. 134

57. The second autonomous flight of *Frog*. Weather conditions were near ideal as *Frog* tracked the 3-D trajectory shown. 135

58. Top graph: Commanded vehicle yaw rate in degrees per second from the controller along the trajectory flown on July 23, 1997. Bottom graph: Lateral error in feet as computed by the guidance algorithm along the same trajectory. 136

59. The output of the longitudinal channel of the controller is shown along with the error signal on which it is acting. Note that the positive z-axis points down and that positive error corresponds to the vehicle being below the reference trajectory. 137

60. Interface called by size_it.m 141

61. Selection of output sensors 142

62. Interface called by view_it.m 158

ACKNOWLEDGMENTS

I would like to thank a few outstanding individuals who embody all that the title *professor* is meant to imply. Professors Kang, Henson and Pascoal made an enormous contribution to my education because they truely love to teach and mentor and for that I am very gratefull. To my fellow cell mate, Osa Fitch, I owe what remains of my sanity. His fine wit made the many late nights and hectic schedules bearable and his keen insight broke many a log jam. A graduate student could not ask for a better advisor, academic mentor or friend then I found in Isaac Kaminer. Simply put, without him, this would not have happened. Of course, life is not simply about aeronautics. I owe an apology to four special children, Nicole, Katie, Eric and Patty who felt the repercussions of the stress and long hours of the program without any idea as to why. Finally, to Tricia, my wife, we made it! I know how you cringed when I told you that I was going to still more school after the last degree. Thank you for enduring all the times that I was not there or only half there lost somewhere in thought. Now, my thoughts are on you.

I. OVERVIEW

A. LAYOUT OF THE CHAPTERS

Methods in integrated plant-controller design and integrated guidance-controller design are contained in three chapters and two appendices. Applicable supporting computer code is explained in the appendices. The organization and content of the body of the work is as follows.

Chapter II addresses the problem of sizing the horizontal tail on a statically unstable transport aircraft. A brief background in linear matrix inequalities prepares the reader for the problem formulation. Recovery from a severe angle-of-attack excursion, or penetration of a vertical wind-shear, is formulated in terms of the solution to a convex minimization problem, and numerically solved. The effects of a number of parameters in the aircraft definition, such as canards and flexible motion, are addressed.

In Chapter III, the problem of unmanned air vehicles tracking inertial trajectories is addressed. The concept of *trimming trajectories* is defined, and the tracking of a trimming trajectory is converted into the problem of driving a generalized error vector - which implicitly includes the distance to the trajectory - to zero. The linearization of the generalized error dynamics along the trajectory is shown to be time-invariant. Using these results, the problem of trajectory tracking is posed and solved within the framework of gain-scheduled control theory. This leads to a new technique for integrated guidance and control system design for unmanned air vehicles.

The development of a rapid prototyping system for flight testing of guidance, navigation, and control algorithms for unmanned air vehicles is presented in Chapter IV. The system affords a small team the ability to take a new concept in guidance, navigation, and control from simulation to flight test. A proof-of-concept flight test demonstration of a new integrated guidance and control algorithm is detailed. The

project includes a parameter identification problem for an unmanned air vehicle at the Naval Postgraduate School. Controller synthesis and implementation follow, based on the identified model. Finally, the system is used to test the theory presented in Chapter III in flight.

B. CONTRIBUTION

This section states the original contribution of this work.

- The problem of sizing a horizontal control surface for a statically unstable transport aircraft is addressed. Closed-loop performance of the plant and controller in response to certain dynamic constraints is used to define a design region that captures the integrated nature of the plant definition and controller synthesis process.
- A numerical tool is developed that determines acceptable combinations of tail volume, center-of-gravity location, and peak actuator rate for a statically unstable transport aircraft using state-feedback, or output-feedback, linear control to recover from a severe angle-of-attack excursion, or vertical wind-shear penetration. The method formulates the dynamic constraint as a convex optimization problem, and uses recently developed LMI solvers to solve the problem numerically.
- Numerical results demonstrate how to quantify the effect of the following in terms of their influence on the sizing of the horizontal control surfaces.
  - The addition of a canard to an aircraft definition.
  - The use of dynamic output versus static, state-feedback linear control.
  - The effect of simple, symmetric aeroelastic motion.
- Work on implementing nonlinear gain-scheduled controllers has been extended to trajectory tracking control problems for unmanned air vehicles. A broad class of trimming trajectories is defined, and a technique is developed that provides independent control of the space and time coordinates along trajectories in the class. Using this technique, it is shown that the performance and robustness of the linear design is recovered locally by the nonlinear plant and gain-scheduled controller.
- A unified system for the synthesis, simulation, and flight testing of avionics algorithms on unmanned air vehicles is constructed. The system is used to take new theory in integrated guidance and control from conception to flight test.

II. INTEGRATED PLANT CONTROLLER OPTIMIZATION

A. INTRODUCTION

The use of an automatic flight augmentation system is commonplace on a modern aircraft. The benefits of its use may include such things as the remedy of undesirable flight characteristics, the reduction of pilot workload, and an increase in performance and fuel efficiency. Whether required for safe flight or not, it is hard to imagine a new transport aircraft being built without a sophisticated flight augmentation system. In fact, the next generation of high-speed civil transport (HSCT) aircraft will require some form of feedback control for safety of flight considerations, since current designs have an unstable short period.

Since HSCT will require some form of automatic flight control always being active, the sizing of its control surfaces is no simple matter. Traditionally, static constraints have been used to size the horizontal tail. For a given tail volume, constraints are calculated that limit the fore and aft travel of the center of gravity. Constraints that limit the forward center-of-gravity position include (1) sufficient nose-up pitch acceleration at the rotation speed (nose-wheel lift off), and (2) sufficient nose-up pitch acceleration at the approach speed in the landing configuration (go-around). Constraints that limit aft center-of-gravity position include (1) at brake release with maximum thrust, sufficient weight on the nose gear (tip back), (2) pitch-up acceleration at the rotation speed (nose-wheel lift off), and (3) sufficient nose-down pitch acceleration at minimum flying speeds [Ref. 26].

At the aft center-of-gravity locations projected for the approach flight condition of the HSCT, dynamic constraints may be more restrictive than static constraints. The need to include dynamic considerations in the configuration definition process has been addressed before. References [Ref. 5] and [Ref. 4] describe early published work by Beaufrere in this area, largely motivated by the X-29 research program. In

[Ref. 41], Schmidt uses the system sensitivity function to describe the fundamental trade-off that exists between the level of static instability that can be controlled and vehicle flexibility. Previous (unpublished) work in industry has used time domain analysis to determine how far aft (how unstable) the center of gravity could be before the airplane was unrecoverable. The analysis included a given design angle-of-attack disturbance, and given rate and position limits of the pitch control effector.

This work extends the work of [Ref. 26], and uses a numerical technique similar to previous work in [Ref. 33]. There an integrated aircraft controller design methodology using LMIs was applied to the control power sizing for an F–14 aircraft. The major contribution of this work is two fold. First, the tail sizing design problem is defined in terms that include the integral participation of a feedback control system. Since the degree of control of the longitudinal dynamics depends on how fast and far the longitudinal control surface(s) can be moved by the control actuator(s), a natural metric that captures the *size* of the automatic flight control system is the maximum actuator rate. The design trade-off naturally includes consideration of actuator performance. For instance, it may be more cost effective to incorporate faster, generally larger and more expensive, actuators rather than pay the drag penalty associated with a larger horizontal tail. In that light, we introduce the following definition.

Tail Sizing Design Space:

> *The Tail Sizing Design Space is the region of "acceptable" combinations of tail volume ($\bar{V}_H$), center-of-gravity station ($x_{c.g.}$), and peak actuator rate ($\dot{u}_{max}$). The triplet $\{\bar{V}_H, x_{c.g.}, \dot{u}_{max}\}$, defines an aircraft model and an automatic flight control system. The model is obtained through the linearization of the nonlinear dynamics of the HSCT at an equilibrium point. It is partially defined by the specified tail volume and center-of-gravity position. The automatic flight control system is characterized by the specified maximum actuator rate in the triplet. By "acceptable" it is meant that, for the model associated with the triplet $\{\bar{V}_H, x_{c.g.}, \dot{u}_{max}\}$, a linear controller is known to exist that 1) stabilizes the plant, 2) meets prescribed dynamic performance constraints, and 3) does not exceed the maximum actuator rate in response to the dynamic constraint.*

Second, a numerical tool is developed that determines the *Tail Sizing Design Space* for a given HSCT dynamic model, flight condition, and dynamic constraint. This tool is termed the *Tail Sizing Design Tool*, and provides the capability to measure the effect of adding a second horizontal control surface in the form of a canard. The *Tail Sizing Design Tool* also provides the capability to measure the effect of simple, symmetric, flexible motion of the vehicle. Based on numerical analysis of designs, conclusions are drawn as to the relative effective of the use of canards, the use of static versus dynamic controllers, and the inclusion of aeroelastic effects.

1. Example of a Design Space

As a motivating example, consider a simple case. Let $a(s)$ and $b(h)$ be two independent variables that are smooth functions of s and h, respectively. Furthermore, assume that they are independent of time, i.e. $\frac{d(a(s))}{dt} = 0$ and $\frac{d(b(h))}{dt} = 0$. Let

$$\frac{dx}{dt} = a(s)x(t) + b(h)u(t), \tag{II.1}$$

where $x, u \in \mathcal{R}^1$. Then, equation II.1 describes a family of dynamic systems. Evaluating (II.1) about arbitrary values of the parameters $a(s)$ and $b(h)$ results in the LTI systems

$$\begin{aligned} \dot{x}(t) &= a(s_0)x(t) + b(h_0)u(t), \\ x(0) &= x_0. \end{aligned} \tag{II.2}$$

It is desired to use feedback of the form

$$u(t) = -k(s_0, h_0)x(t) \tag{II.3}$$

to stabilize (II.2). Lyapunov stability theory states that this is equivalent to the existence of a $p > 0$ such that

$$2p\{a(s_0) - b(h_0)k(s_0, h_0)\} < 0. \tag{II.4}$$

This implies that

$$k(s_0, h_0) > \frac{a(s_0)}{b(h_0)} \tag{II.5}$$

guarantees local stability of (II.1) at (s_0, h_0). This places a lower bound on $k(s_0, h_0)$.

Assume that the absolute value of $u(t)$ is constrained to be less than some nominal value, u_{max}. For this simple system, the maximum value of $|u(t)|$ occurs at $t = 0$ and is equal to

$$u_{max} = |k(s_0, h_0)x_0|. \tag{II.6}$$

This places an upper bound on $k(s_0, h_0)$,

$$k(s_0, h_0) < \frac{u_{max}}{x_0}. \tag{II.7}$$

The two limits, (II.5) and (II.6), are shown graphically in Figure 1.

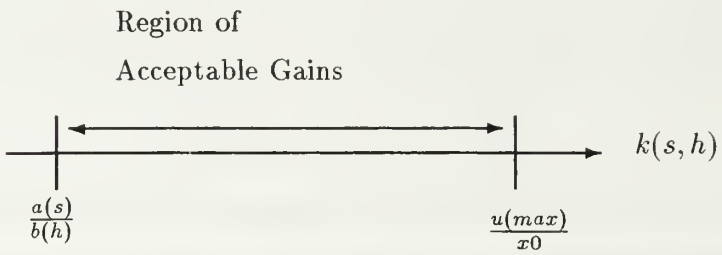


Figure 1. Range of Feedback Gains

If the boundaries, $\frac{a(s_0)}{b(h_0)}$, $\frac{u_{max}}{x_0}$, are allowed to approach each other, the region of acceptable feedback gains approaches a single point where

$$\frac{a(s_0)}{b(h_0)} = \frac{u_{max}}{x_0}. \tag{II.8}$$

Suppose $a(s)$ varies linearly from a minimal value of 0.1 to a maximum value of 1.0. Suppose $b(h)$ varies linearly from a minimal value of 0.3 to a maximum value of 0.9. A value of 1 was used for x_0. The family of LTI systems, which validate the relationship in expression II.8, is represented by the curved surface shown in Figure 2.

Below the surface, the LTI systems are not stabilizable subject to the stated constraint. Above the surface, multiple controllers exist that stabilize a given system subject to the stated constraint. If the maximum acceptable value of $u(t)$ is included

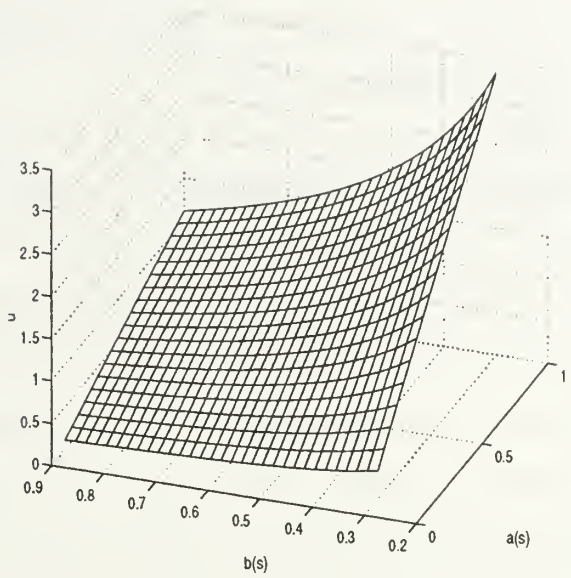


Figure 2. Family of LTI systems with a single acceptable gain.

as a horizontal plane, the *Design Space* for this simple triplet, $\{a, \ , b \ , u_{max}\}$, consists of the region above the surface in Figure 2 and below this horizontal plane. This is shown in Figure 3.

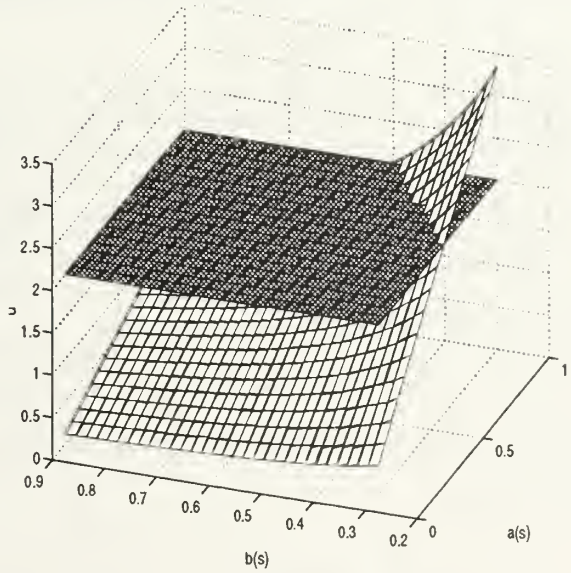


Figure 3. Example of a *Design Space*

The intersection of the two surfaces is a line which represents limiting combinations of $a(s)$ and $b(s)$ for a fixed u_{max} and x_0.

The chapter begins with a background in Linear Matrix Inequalities sufficient to formulate the general problem in a form suitable for solution numerically. Next, the *Tail Sizing Design Tool* is developed for a rigid-body HSCT model. The dynamic constraint associated with the problem is a severe angle-of-attack disturbance. Numerical results follow based on an application of the *Tail Sizing Design Tool* to an HSCT model representative of current designs. Then, a method is developed that models the effects of simple, symmetric, flexible motion of the HSCT. Numerical results follow based on applying the *Tail Sizing Design Tool* to an aeroelastic HSCT model. Comparisons to the rigid-body model are made. Finally, the *Tail Sizing Design Tool* is developed for a rigid-body HSCT model where the dynamic constraint is the penetration of a vertical wind-shear. The chapter ends with conclusions and recommendations.

B. BACKGROUND IN LINEAR MATRIX INEQUALITIES

It is usually not possible to express the solution to a complex problem with multiple constraints analytically. Progress in computational power, however, make numerical solutions to these problems attractive. Efficient convex optimization algorithms, recently made available from *The Math Works* [Ref. 17], allow problems that can be formulated in terms of LMIs to be solved exactly.

Two major results in LMI theory are drawn on to formulate the problem of sizing a longitudinal control surface. The first concept is that of an invariant ellipsoid.

Consider an LTI system:

$$\begin{aligned} \dot{x} &= Fx, \\ z &= Gx, \end{aligned} \tag{II.9}$$

where $x \in \mathcal{R}^n$, $z \in \mathcal{R}^p$. Let $P > 0$ and define

$$\mathcal{E} = \{\zeta \in \mathcal{R}^n : \zeta^T P \zeta \leq 1\}.$$

Then $\mathcal{E}$ is called an invariant ellipsoid associated with (II.9) if, for every trajectory x satisfying (II.9), $x(0) \in \mathcal{E}$ implies $x(t)$ is in $\mathcal{E}$, for all $t \geq 0$ [Ref. 6].

Let $\mathcal{E}$ be an invariant ellipsoid for (II.9), and define $V(x) = x^T P x$. From the definition of $\mathcal{E}$, it follows that $V(x) \leq V(x(0)) \leq 1$, and

$$\begin{aligned} \dot{V}(x) &= x^T(F^T P + PF)x \leq 0, \\ &\Rightarrow \quad F^T P + PF \leq 0. \end{aligned}$$

On the other hand, suppose $\exists P > 0$ such that $F^T P + PF \leq 0$ and $x(0)^T P x(0) \leq 1$. Then, for $x(t)$ satisfying (II.9), we obtain

$$x^T(F^T P + PF)x =: \dot{V}(x) \leq 0.$$

Integrating both sides from 0 to $T \geq 0$ we get

$$\begin{aligned} V(x(T)) - V(x(0)) &\leq 0, \\ &\Rightarrow \quad V(x(T)) \leq V(x(0)) \leq 1, \end{aligned}$$

for any $T \geq 0$. Therefore, we have shown that $\mathcal{E}$ is an invariant ellipsoid associated with the linear system (II.9) if and only if $F^T P + PF \leq 0$.

Next, we will show how the idea of invariant ellipsoids can be used to obtain bounds on the peak of the output in response to a known initial condition. Let $\mathcal{E}$ be an invariant ellipsoid associated with the linear system (II.9). Then,

$$\|z(t)\|^2 = z(t)^T z(t) \leq \max_{\zeta \in \mathcal{E}} \zeta^T G^T G \zeta \qquad \forall t \geq 0.$$

Now,

$$\begin{aligned} \max_{\zeta \in \mathcal{E}} \zeta^T G^T G \zeta &= \max_{\|u\| \leq 1} u^T P^{-1/2} G^T G P^{-1/2} u, \\ &= \|G P^{-1/2}\|^2, \\ &= \lambda_{\max}(P^{-1/2} G^T G P^{-1/2}), \end{aligned} \tag{II.10}$$

where $\lambda_{\max}(.)$ denotes the maximum eigenvalue of the argument. Therefore, an upper bound on the peak of $\|z(t)\|$ in response to $x(0)$ can be found as a square root of the minimum of δ subject to:

$$\delta - \lambda_{max}(P^{-1/2}G^T G P^{-1/2}) \geq 0, \tag{II.11}$$

$$x(0)^T P x(0) \leq 1, \tag{II.12}$$

$$F^T P + PF \leq 0. \tag{II.13}$$

Nonlinear inequalities, such as equation II.27, can be converted to LMI form using Schur complements [Ref. 22]. In general, any nonlinear inequality of the form

$$R > 0, \qquad Q - SR^{-1}S^T > 0, \tag{II.14}$$

for symmetric Q and R, is equivalent to the LMI

$$\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} > 0. \tag{II.15}$$

To see this, consider the congruence transformation,

$$\begin{bmatrix} I & -SR^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} I & 0 \\ -R^{-T}S^T & I \end{bmatrix} = \begin{bmatrix} Q - SR^{-1}S^T & 0 \\ 0 & R \end{bmatrix}, \tag{II.16}$$

where the block diagonal structure makes equivalence to expression II.14 obvious. Using Schur complements on expression II.27 yields,

$$\delta - \lambda_{max}(P^{-1/2}G^T G P^{-1/2}) \geq 0 \iff$$

$$\delta I - P^{-1/2}G^T G P^{-1/2} \geq 0 \iff$$

$$\begin{bmatrix} \delta & GP^{-1/2} \\ P^{-1/2}G^T & I \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & P^{-1/2} \end{bmatrix} \begin{bmatrix} \delta & G \\ G^T & P \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & P^{-1/2} \end{bmatrix} \geq 0 \iff$$

$$\begin{bmatrix} \delta & G \\ G^T & P \end{bmatrix} \geq 0 \iff \begin{bmatrix} P & G^T \\ G & \delta \end{bmatrix} \geq 0.$$

In summary, the square root of a solution to the following optimization problem:

$$\begin{aligned} & min \ \delta \\ & \text{subject to} \\ & \begin{bmatrix} P & G^T \\ G & \delta \end{bmatrix} \geq 0, && \text{(II.17)} \\ & x(0)^T P x(0) \leq 1, && \text{(II.18)} \\ & F^T P + PF \leq 0, && \text{(II.19)} \\ & P > 0, && \text{(II.20)} \end{aligned}$$

provides an upper bound on the peak of $\|z(t)\|$ for the system (II.9) in response to the initial condition $x(0)$.

The second concept used is that of an LMI region. In their recent work, Gahinet and Chilali [Ref. 7] introduce the concept of an LMI region, and have shown that such regions can be characterized by LMIs. LMI regions include sectors, disks, conic sectors, strips, as well as the intersection of any of these. For our purposes, the intersection of just two regions will suffice. The first region is a conic sector centered at the origin with half inner angle ϕ. The second region is the left half-plane with its right boundary at $-\beta$. The intersection of these regions is shown as the region $\mathcal{L}$ in Figure 4.

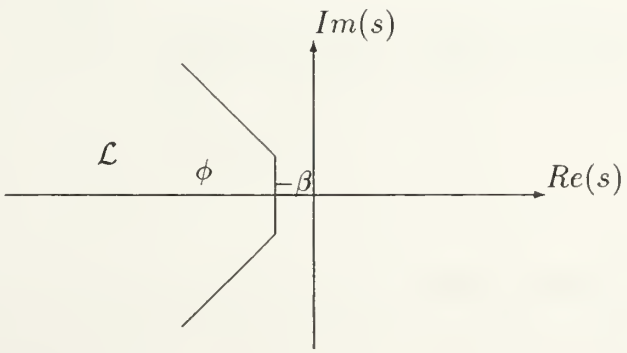


Figure 4. Region $\mathcal{L}$

The eigenvalues of the system described by equation II.9 lie in the region $\mathcal{L}$, if and only if there exists a positive definite matrix P, such that

$$\begin{bmatrix} \sin\phi(F^TP + PF) & \cos\phi(-PF + F^TP) & 0 \\ \cos\phi(PF - F^TP) & \sin\phi(F^TP + PF) & 0 \\ 0 & 0 & F^TP + PF + P2\beta \end{bmatrix} < 0. \quad \text{(II.21)}$$

1. State Feedback Synthesis LMIs

A feedback interconnection of an LTI system and state-feedback controller is

$$\left\{ \begin{array}{lll} \dot{x} & = Ax + Bu & \\ z & = Cx + Du & = \left\{ \begin{array}{ll} \dot{x} & = [A + BK]x \\ z & = [C + DK]x, \end{array} \right. \\ u & = Kx & \end{array} \right. \quad \text{(II.22)}$$

where $x \in \mathcal{R}^n$, $u \in \mathcal{R}^m$, $z \in \mathcal{R}^p$. Substitution into the previous invariant ellipsoid and pole placement region matrix inequalities result in:

$$\begin{array}{rl} \min & z_{max} \\ \text{subject to} & \end{array}$$

$$\begin{bmatrix} P & (C + DK)^T \\ (C + DK) & z_{max}^2 \end{bmatrix} \geq 0,$$

$$x(0)^T P x(0) \leq 0,$$

$$\begin{bmatrix} (\sin\phi)(A^TP + K^TB^TP + PA + PBK) & (\cos\phi)(-PA - PBK + A^TP + K^TB^TP) & \\ (\cos\phi)(PA + PBK - A^TP - K^TB^TP) & (\sin\phi)(A^TP + K^TB^TP + PA + PBK) & \cdots \\ 0 & 0 & \\ & & \begin{array}{c} 0 \\ 0 \\ A^TP + K^TB^TP + PA + PBK + P2\beta \end{array} \end{bmatrix} \leq 0,$$

$$P > 0. \quad \text{(II.23)}$$

The above matrix inequalities are no longer affine in the unknown parameters, since the state-feedback controller, K, and positive definite matrix, P, appear as multiplicative terms. Since P is positive definite, let $Y = P^{-1}$. Perform the following congruence transformations, and perform the substitution of variables, $W = KY$.

$$\begin{bmatrix} P & (C+DK)^T \\ (C+DK) & z_{max}^2 \end{bmatrix} \geq 0 \Leftrightarrow$$

$$\begin{bmatrix} Y & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} P & (C+DK)^T \\ K & z_{max}^2 \end{bmatrix} \begin{bmatrix} Y & 0 \\ 0 & I \end{bmatrix} \geq 0 \Leftrightarrow$$

$$\begin{bmatrix} Y & ((C+DK)Y)^T \\ (C+DK)Y & z_{max}^2 \end{bmatrix} \geq 0 \Leftrightarrow$$

$$\begin{bmatrix} Y & (CY+DW)^T \\ (CY+DW) & z_{max}^2 \end{bmatrix} \geq 0. \qquad \text{(II.24)}$$

$$\begin{bmatrix} Y & 0 & 0 \\ 0 & Y & 0 \\ 0 & 0 & Y \end{bmatrix} \begin{bmatrix} (\sin\phi)(A^TP + K^TB^TP + PA + PBK) & (\cos\phi)(-PA - PBK + A^TP + K^TB^TP) & \\ (\cos\phi)(PA + PBK - A^TP - K^TB^TP) & (\sin\phi)(A^TP + K^TB^TP + PA + PBK) & \cdots \\ 0 & 0 & \\ & 0 & \\ & 0 & \\ & A^TP + K^TB^TP + PA + PBK + P2\beta & \end{bmatrix} \begin{bmatrix} Y & 0 & 0 \\ 0 & Y & 0 \\ 0 & 0 & Y \end{bmatrix} \leq 0 \iff$$

$$\begin{bmatrix} (\sin\phi)(YA^T + W^TB^T + AY + BW) & (\cos\phi)(-AY - BW + YA^T + W^TB^T) & \\ (\cos\phi)(AY + BW - YA^T - W^TB^T) & (\sin\phi)(YA^T + W^TB^T + AY + BW) & \cdots \\ 0 & 0 & \\ & 0 & \\ & 0 & \\ & YA^T + W^TB^T + AY + BW + Y2\beta & \end{bmatrix} \leq 0. \qquad \text{(II.25)}$$

Using Schur complements:

$$\begin{aligned} x(0)^TPx(0) &\leq 1 \iff \\ x(0)^TY^{-1}x(0) &\leq 1 \iff \\ \begin{bmatrix} 1 & x(0)^T \\ x(0) & Y \end{bmatrix} &\geq 0. \end{aligned} \qquad \text{(II.26)}$$

Once again, the expressions II.24, II.25, II.26 constitute an LMI in the unknown matrix variables, W and Y. Suppose $\exists\, Y > 0$ and W such that they validate expressions II.24, II.25, II.26. Then, $K = WY^{-1}$ is the state-feedback controller that maintains $\|z\| < z_{max}$ in response to the initial condition $x(0), \forall t \geq 0$.

2. Output Feedback Synthesis LMIs

A feedback interconnection of an LTI system and strictly proper, output-feedback controller is

$$\left\{\begin{array}{lll} \dot{x} &= Ax + Bu & \\ y &= Cx & \\ z &= C_z x + D_z u & = \left\{\begin{array}{lll} \dot{\eta} &= \left[\begin{array}{cc} A & BC_k \\ B_k C & A_k \end{array}\right]\eta \\ z &= \left[\begin{array}{cc} C_z & D_z D_k \end{array}\right]\eta, \end{array}\right. \\ \dot{x}_k &= A_k x_k + B_k y & \\ u &= C_k x_k & \end{array}\right. \tag{II.27}$$

where $x \in \mathcal{R}^n, x_k \in \mathcal{R}^n$, $u \in \mathcal{R}^m$, $z \in \mathcal{R}^p$, $y \in \mathcal{R}^j$, $\eta \in \mathcal{R}^{2n}$, and where $\eta^T = [x^T \; x_k^T]$. The initial condition is $\eta(0)^T = \eta_0^T = [x_0^T \; x_{k_0}^T]$.

The four analysis matrix inequalities based on using the concept of an invariant ellipsoid and pole placement region are gathered for convenience.

$$\left[\begin{array}{cc} P & G^T \\ G & \delta I \end{array}\right] \geq 0, \tag{II.28}$$

$$P > 0, \tag{II.29}$$

$$x_0^T P x_0 \leq 1, \tag{II.30}$$

$$\left[\begin{array}{ccc} (\sin\phi)(F^T P + PF) & (\cos\phi)(-PF + F^T P) & 0 \\ (\cos\phi)(PF - F^T P) & (\sin\phi)(F^T P + PF) & 0 \\ 0 & 0 & F^T P + PF + P2\beta \end{array}\right] \leq 0. \tag{II.31}$$

If the closed-loop system matrices in expression II.27 were substituted into expressions II.28, II.29, II.30, and II.31, the expressions would not be affine in the unknown matrix variables. Similar to the state-feedback case, the key is to find a congruence transformation with a "linearizing" effect on the unknown matrix variables. In a recent work by Scherer and Gahinet [Ref. 8], the authors prove that such

a transformation exists for the output feedback case. Consider the following partition of P and P^{-1},

$$P = \begin{bmatrix} S & N \\ N^T & Q \end{bmatrix}, \tag{II.32}$$

$$P^{-1} = \begin{bmatrix} R & M \\ M^T & V \end{bmatrix}. \tag{II.33}$$

Since P and P^{-1} are both symmertic, so are R and S. From the identity

$$\begin{aligned} P^{-1}P &= \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \\ &= \begin{bmatrix} R & M \\ M^T & V \end{bmatrix} \begin{bmatrix} S & N \\ N^T & Q \end{bmatrix}, \\ &= \begin{bmatrix} RS + MN^T & RN + MQ \\ M^TS + VN^T & M^TN + VQ \end{bmatrix}, \end{aligned} \tag{II.34}$$

we can infer

$$RS + MN^T = I,$$

or

$$NM^T = I - SR. \tag{II.35}$$

Also,

$$P^{-1} \begin{bmatrix} S \\ N^T \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix}, \tag{II.36}$$

and

$$P \begin{bmatrix} R \\ M^T \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{II.37}$$

These identities are used to define the key matrices used in the congruence transformation as follows:

$$P\begin{bmatrix} R & I \\ M^T & 0 \end{bmatrix} = \begin{bmatrix} I & S \\ 0 & N^T \end{bmatrix}, \\ =: X_2, \tag{II.38}$$

and

$$P^{-1}\begin{bmatrix} I & S \\ 0 & N^T \end{bmatrix} = \begin{bmatrix} R & I \\ M^T & 0 \end{bmatrix}, \\ =: X_1. \tag{II.39}$$

Notice that

$$\begin{aligned} X_2X_1^{-1} &= \begin{bmatrix} I & S \\ 0 & N^T \end{bmatrix}\begin{bmatrix} I & S \\ 0 & N^T \end{bmatrix}^{-1} P, \\ &= P. \end{aligned} \tag{II.40}$$

Using X_1 and X_2, consider the following congruence transformation:

$$\begin{bmatrix} X_1^T & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} P & G^T \\ G & \delta I \end{bmatrix}\begin{bmatrix} X_1 & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} X_1^TPX_1 & X_1^TG^T \\ GX_1 & \delta I \end{bmatrix}. \tag{II.41}$$

Expanding the (1,1) block of expression II.41, we obtain

$$\begin{aligned} X_1^TPX_1 &= X_1^TX_2X_1^{-1}X_1, \\ &= X_1^TX_2, \\ &= \begin{bmatrix} R & M \\ I & 0 \end{bmatrix}\begin{bmatrix} I & S \\ 0 & N^T \end{bmatrix}, \\ &= \begin{bmatrix} R & I \\ I & S \end{bmatrix}. \end{aligned} \tag{II.42}$$

Expanding the (2,1) block of expression II.41 for the general case of output feedback, we obtain

$$\begin{aligned} GX_1 &= \begin{bmatrix} C_z & D_z C_k \end{bmatrix} \begin{bmatrix} R & I \\ M^T & 0 \end{bmatrix}, \\ &= \begin{bmatrix} C_z R + D_z C_k M^T & C_z \end{bmatrix}. \end{aligned} \tag{II.43}$$

The matrix inequality in II.28 becomes

$$\begin{bmatrix} R & I & MC_k^T D_z^T + R^T C_z^T \\ I & S & C_z^T \\ C_z R + D_z C_k M^T & C_z & z_{max}^2 \end{bmatrix} \geq 0. \tag{II.44}$$

After a congruence transformation with X_1, the LMI in (II.29) becomes

$$\begin{bmatrix} R & I \\ I & S \end{bmatrix} \geq 0. \tag{II.45}$$

This LMI is accounted for as a principle sub-matrix of the LMI in (II.44). Schur complements are used to express the LMI in (II.30) as

$$\begin{bmatrix} 1 & \eta_0^T \\ \eta_0 & P^{-1} \end{bmatrix} \geq 0. \tag{II.46}$$

Assume $x_{k_0} = 0$, and consider the following congruence transformation:

$$\begin{aligned} \begin{bmatrix} 1 & 0 \\ 0 & X_2^T \end{bmatrix} \begin{bmatrix} 1 & \eta_0^T \\ \eta_0 & P^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & X_2 \end{bmatrix} &= \begin{bmatrix} 1 & \eta_0^T X_2 \\ X_2^T \eta_0 & X_2^T P^{-1} X_2 \end{bmatrix}, \\ &= \begin{bmatrix} 1 & \eta_0^T X_2 \\ X_2^T \eta_0 & X_2^T X_1 \end{bmatrix}. \end{aligned} \tag{II.47}$$

Expanding the (1,2) block of equation II.47 yields,

$$\eta_0^T \begin{bmatrix} I & S \\ 0 & N^T \end{bmatrix} = \begin{bmatrix} x_0^T & x_0^T S \end{bmatrix}. \tag{II.48}$$

After the appropriate congruence transformation, and substituting (II.48) into (II.47), we have

$$\begin{bmatrix} 1 & x_0^T & x_0^T S \\ x_0 & R & I \\ S^T x_0 & I & S \end{bmatrix} \geq 0. \tag{II.49}$$

To see how the pole placement region LMI is transformed, look at the transformation of the (1,1) block of equation II.31.

$$\begin{aligned} F^T P + PF &< 0 \iff \\ P^{-1} F^T + F P^{-1} &< 0 \iff \\ X_2^T P^{-1} F^T X_2 + X_2^T F P^{-1} X_2 &< 0. \end{aligned} \tag{II.50}$$

Since $P^{-1} = X_1 X_2^{-1}$ and $P^{-T} = X_2^{-T} X_1^T$, equation II.50 becomes

$$X_1^T F^T X_2 + X_2^T F X_1 < 0. \tag{II.51}$$

For output feedback synthesis

$$F = \begin{bmatrix} A & BC_k \\ B_k C & A_k \end{bmatrix}. \tag{II.52}$$

Substituting (II.52) into the first term in (II.51) results in

$$X_2^T F X_1 = \begin{bmatrix} I & 0 \\ S & N \end{bmatrix} \begin{bmatrix} A & BC_k \\ B_k C & A_k \end{bmatrix} \begin{bmatrix} R & I \\ M^T & 0 \end{bmatrix},$$

$$= \begin{bmatrix} A & BC_k \\ SA + NB_kC & SBC_k + NA_k \end{bmatrix} \begin{bmatrix} R & I \\ M^T & 0 \end{bmatrix},$$

$$= \begin{bmatrix} AR + BC_kM^T & A \\ SAR + NB_kCR + SBC_kM^T + NA_kM^T & SA + NB_kC \end{bmatrix}. \tag{II.53}$$

Define the change of variables,

$$\bar{C}_k = C_kM^T, \tag{II.54}$$

$$\bar{B}_k = NB_kC, \tag{II.55}$$

$$\bar{A}_k = SAR + NB_kCR + SBC_kM^T + NA_kM^T, \tag{II.56}$$

and equation II.53 is rewritten as

$$\begin{bmatrix} AR + B\bar{C}_k & A \\ \bar{A}_k & SA + \bar{B}_k \end{bmatrix} =: \Xi. \tag{II.57}$$

Each term in the pole placement LMI is transformed in a similar fashion. This results in

$$\begin{bmatrix} \sin\phi(\Xi^T + \Xi) & \cos\phi(-\Xi + \Xi^T) & 0 \\ \cos\phi(\Xi - \Xi^T) & \sin\phi(\Xi^T + \Xi) & 0 \\ 0 & 0 & \Xi^T + \Xi + \begin{bmatrix} R & I \\ I & S \end{bmatrix} 2\beta \end{bmatrix} \leq 0. \tag{II.58}$$

Notice that expressions II.49, II.45, II.44, and II.58 are affine in the unknown matrix variables $\bar{A}_k$, $\bar{B}_k$, $\bar{C}_k$, R, and S. Thus, they constitute an LMI. Given $\bar{A}_k$, $\bar{B}_k$, $\bar{C}_k$, R, and S that validate expressions II.49, II.45, II.44, and II.58, the output-feedback controller is reconstructed as follows. First, reconstruct the matrices M and N via a singular value decomposition.

$$U\Sigma^2 V^T = I - RS, \tag{II.59}$$

$$N^{-1} = \Sigma V, \tag{II.60}$$

$$M^{-T} = U\Sigma. \tag{II.61}$$

Form the controller by inverting the change of variables in expressions II.55, II.56, and II.56 as

$$A_k = N^{-1}(\bar{A}_k - SA_pR - \bar{B}_kC_pR - SB_p\bar{C}_k)M^{-T}, \tag{II.62}$$

$$B_k = N^{-1}\bar{B}_k, \tag{II.63}$$

$$C_k = \bar{C}_kM^{-T}. \tag{II.64}$$

C. HIGH ANGLE OF ATTACK RECOVERY

1. Problem Formulation for a Rigid Body HSCT

The *Plant Controller Optimization* (PCO) problem to be solved in this section can be stated as follows:

> *Let an HSCT flight condition be specified by the aircraft's flight speed, altitude, and flight path angle. Furthermore, let a certain set of flying quality requirements exist for the HSCT at that flight condition. Let the dynamic constraint be defined as the recovery of the aircraft from a high angle-of-attack excursion, while not exceeding certain peak actuator rate and actuator amplitude limits. Then, for a range of horizontal tail volumes, and for a range of peak actuator rates, determine the aft center-of-gravity limits for which there still exists a linear feedback controller that 1) stabilizes the plant, 2) satisfies the flying quality requirements, 3) satisfies the dynamic constraint.*

Let x_{cg} denote the center-of-gravity location as a fraction of the reference chord, and let x denote the vector of HSCT longitudinal states. Let u denote the horizontal tail incidence angle, and let $\bar{V}_H$ denote the tail volume. Figure 5 provides a description of the tail volume, $\bar{V}_H$.

This representation is convenient since, for one flight condition, the wing-body mean aerodynamic center remains constant. Thus, the control surface volume and

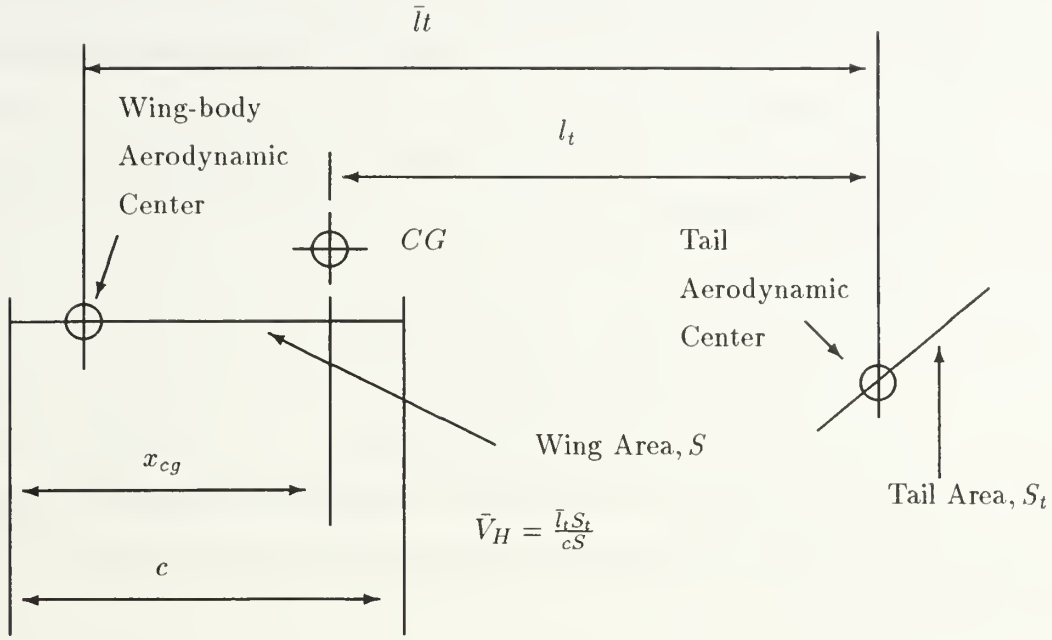


Figure 5. Definition of $\bar{V}_H$.

center of gravity are decoupled in terms of their influence on the vehicle's dynamics. The longitudinal dynamics of the aircraft can be expressed by the following system of differential equations:

$$\mathcal{G} = \begin{cases} \dot{x} &= \mathcal{F}(x, u, x_{cg}, \bar{V}_H), \\ z &= \mathcal{H}(x, u), \end{cases} \tag{II.65}$$

where $\mathcal{F}(.)$ is a C^1 function of $x, u, x_{cg}, \bar{V}_H$, $\mathcal{H}$ is a C^1 function relating x and z, and $z = [V_T \ \gamma]^T$ defines the true airspeed (V_T) and flight path angle (γ). Let x_0, u_0 denote the trim values of x and u for a given z_0 and center-of-gravity location, x_{cg_0}, i.e. $\mathcal{F}(x_0, u_0, x_{cg_0}, \bar{V}_{H_0}) = 0$ and $\mathcal{H}(x_0, u_0) = z_0$. Here the vector z is used to characterize the flight condition. Later, when x_{cg} is allowed to change, the trim values, x_0 and u_0, will be recomputed for given values of x_{cg_0}, z_0 and $\bar{V}_{H_0}$. Now, linearizing $\mathcal{G}$ about x_0, u_0 yields a system of linear differential equations,

$$\delta\dot{x} = A(z_0, x_{cg_0}, \bar{V}_{H_0})\delta x + B(z_0, x_{cg_0}, \bar{V}_{H_0})\delta u, \tag{II.66}$$

where δx and δu denote small perturbations in x and u about x_0 and u_0, respectively. In the numerical analysis, the velocity components of δx were normalized by the trim value of the true airspeed.

Let the actuator dynamics be independent of $(Z_0, x_{cg_0}, \bar{V}_{H_0})$, and let them be given by the set of differential equations,

$$\begin{aligned} \delta\dot{x}_a &= A_a\delta x_a + B_a\delta\bar{u}, \\ \delta u &= C_a\delta x_a, \\ \delta\dot{u} &= C_r\delta x_a + D_r\delta\bar{u}, \end{aligned} \tag{II.67}$$

where δu denotes the actuator amplitude and $\delta\dot{u}$ denotes the actuator rate. Appending the actuator dynamics in series with the linearized longitudinal dynamics results in the system:

$$\mathcal{G}_l\,(z_0, x_{cg_0}, \bar{V}_{H_0}) = \left\{ \begin{aligned} \delta\dot{\nu} &= \begin{bmatrix} A & BC_a \\ 0 & A_a \end{bmatrix}\delta\nu + \begin{bmatrix} 0 \\ B_a \end{bmatrix}\delta\bar{u} \\ \delta u &= \begin{bmatrix} 0 & C_a \end{bmatrix}\delta\nu \\ \delta\dot{u} &= \begin{bmatrix} 0 & C_r \end{bmatrix}\delta\nu + D_r\delta\bar{u}, \end{aligned} \right. \tag{II.68}$$

where

$$\delta\nu^T = \begin{bmatrix} \delta x^T & \delta x_a^T \end{bmatrix}. \tag{II.69}$$

Flying quality requirements are typically characterized by the level of attention and skill required of the pilot to control the aircraft. They are grouped in three levels. A lower level corresponds to more benign flight characteristics. In order to achieve certain Level II flying qualities requirements, the eigenvalues of $\mathcal{G}_l$ must be placed in a more restrictive region in the left half plane. Figure 6 shows suggested locations of the Category B, closed-loop pole locations of the HSCT short-perod mode. These locations meet Level II flying quality requirements for the flight condition used in this study. Notice, these locations can be characterized as having a minimum damping ratio of 0.2 and natural frequency of 0.2 radians per second. This suggests that, in

order to meet Level II requirements for the HSCT, the short-period eigenvalues of the closed-loop system $\mathcal{G}_l$ must be placed in the region in the complex plane given by Figure 7.

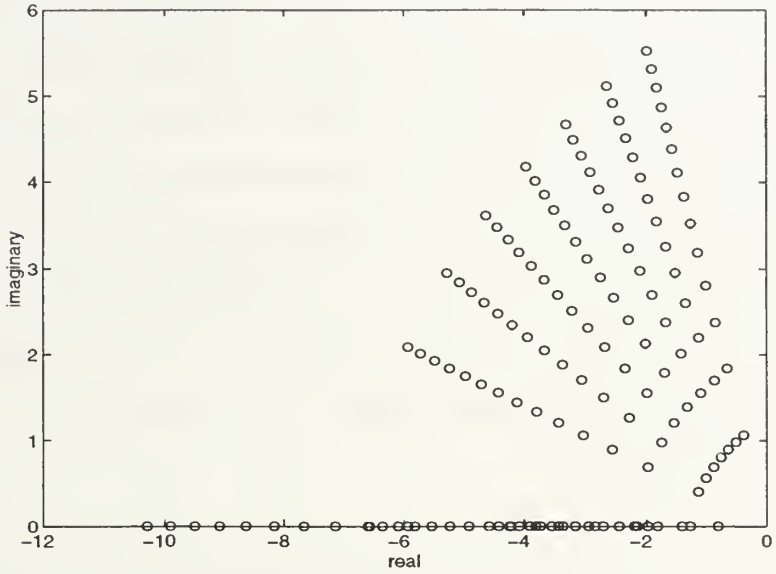


Figure 6. Acceptable Short Period Pole Locations

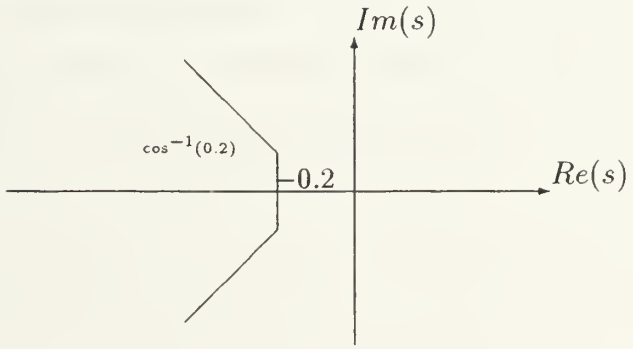


Figure 7. Region $\mathcal{L}$ for Level II Flying Qualities

2. Canard Configuration

The horizontal tail is the most significant control effector for longitudinal control of the aircraft. It is not uncommon, however, for long, slender aircraft designs to include canards for supplemental longitudinal control. The XB-70 and B-1B are two examples. Their presence is usually attributed to flying quality control issues involving the flexible nature of these aircraft [Ref. 1]. This point is addressed in a subsequent section concerning control of an aeroelastic aerodynamic model. To lay the ground work for that section, and as a baseline point of comparison for a rigid structure, this section addresses the addition of a longitudinal control effector forward of the wing. The effect on the *Tail Sizing Design Space* is explored when the feedback control system is free to utilize both control surfaces, in order to recover from angle-of-attack excursions.

The revisions to the nonlinear equations of motion required to account for the addition of the canard parallel the development in [Ref. 14]. When these nonlinear equations of motion are linearized about the equilibrium point determined by $(z_0, x_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0})$, the following LTI system results,

$$\delta\dot{x} = A(z_0, x_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0})\delta x + B_c(z_0, x_{cg_0}, \bar{V}_{C_0})\delta u_c + B_h(z_0, x_{cg_0}, \bar{V}_{H_0})\delta u_h, \qquad \text{(II.70)}$$

where u_c and u_h represent movement of the canard and horizontal tail, respectively, and $\bar{V}_{C_0}$ is the canard volume. Actuator dynamics for both the canard and horizontal tail are appended to the linearized longitudinal dynamics as before. The actuator dynamics are given by:

$$
\begin{aligned}
\delta\dot{x}_a &= \begin{bmatrix} A_{a_c} & 0 \\ 0 & A_{a_h} \end{bmatrix} \delta x_a + \begin{bmatrix} B_{a_c} & 0 \\ 0 & B_{a_h} \end{bmatrix} \begin{bmatrix} \delta\bar{u}_c \\ \delta\bar{u}_h \end{bmatrix}, \\
\delta x_a &= \begin{bmatrix} \delta x_{a_c}^T & \delta x_{a_h}^T \end{bmatrix}^T, \\
\delta u_c &= C_{a_c} \delta x_a, \\
\delta u_h &= C_{a_h} \delta x_a, \\
\delta\dot{u}_c &= C_{r_c} \delta x_a + D_{r_c} \delta\bar{u}_c, \\
\delta\dot{u}_h &= C_{r_h} \delta x_a + D_{r_h} \delta\bar{u}_h.
\end{aligned}
\tag{II.71}
$$

This results in the following system,

$$
\mathcal{G}_l\left(z_0, x_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0}\right) = \left\{
\begin{aligned}
\delta\dot{\nu} &= \begin{bmatrix} A & B_c C_{a_c} & B_h C_{a_h} \\ 0 & A_{a_c} & 0 \\ 0 & 0 & A_{a_h} \end{bmatrix} \delta\nu + \begin{bmatrix} B_{a_c} & 0 \\ 0 & B_{a_h} \end{bmatrix} \begin{bmatrix} \delta\bar{u}_c \\ \delta\bar{u}_h \end{bmatrix} \\
\delta u_c &= \begin{bmatrix} 0 & C_{a_c} & 0 \end{bmatrix} \delta\nu \\
\delta\dot{u}_c &= \begin{bmatrix} 0 & C_{r_c} & 0 \end{bmatrix} \delta\nu + \begin{bmatrix} D_{r_c} & 0 \end{bmatrix} \begin{bmatrix} \delta\bar{u}_c \\ \delta\bar{u}_h \end{bmatrix} \\
\delta u_h &= \begin{bmatrix} 0 & 0 & C_{a_h} \end{bmatrix} \delta\nu \\
\delta\dot{u}_h &= \begin{bmatrix} 0 & 0 & C_{r_h} \end{bmatrix} \delta\nu + \begin{bmatrix} 0 & D_{r_h} \end{bmatrix} \begin{bmatrix} \delta\bar{u}_c \\ \delta\bar{u}_h \end{bmatrix},
\end{aligned}
\right.
\tag{II.72}
$$

where

$$
\delta\nu = \begin{bmatrix} \delta x^T & \delta x_{a_c}^T & \delta x_{a_h}^T \end{bmatrix}^T.
$$

The initial condition is defined by the angle-of-attack perturbation. Let the angle-of-attack perturbation be given as α_0; then the initial condition is

$$
\delta\nu_0 = \begin{bmatrix} 0 & V_t \cos^{-1}(\alpha_0) & 0 & 0 & 0 & 0 \end{bmatrix}^T.
$$

For each design point, $\mathcal{G}_l\left(z_0, x_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0}\right)$, the question becomes, is the set of feedback controllers that recover the aircraft from the angle-of-attack excursion, while

maintaining acceptable flying qualities, and without saturating the actuator, or exceeding a certain actuator rate, empty?

3. Proposed Numerical Solution

In this section, we make the conjecture that the problem at hand is analogous to the example presented in the introductory section. The center-of-gravity location plays the role of the parameter influencing the stability of the open-loop plant. The tail volume plays the role of the parameter influencing how controllable the plant is from the input where feedback is to be applied. It is shown that when the plant parameters are fixed, the problem can be formulated in terms of an LMI, which is affine in both the controller parameters and actuator limits. *Minimizing the actuator rate limit is a convex optimization problem that can be solved exactly.* For one set of plant parameters, let the result of the process be characterized by the final values of all of the parameters. Repeat the process for different sets of values of the plant parameters. Then, an assumption is made that the points lie on a smooth hypersurface. If the grid of plant parameters is sufficiently fine, the shape of the hypersurface can be discerned. The projection of the hypersurface into the three dimensional space spanned by tail volume, center-of-gravity location, and actuator rate limit defines a surface in the *Tail Sizing Design Space* that is extremely useful in the design of the aircraft.

Let

$$
\begin{aligned}
&\Phi_1(\mathcal{G}_l\,(Z_0, x_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0}), u_{h_{max}}, \dot{u}_{h_{max}}, u_{c_{max}}, \dot{u}_{c_{max}}, \nu_0) = \\
&\{W, Y > 0 : \\
&\begin{bmatrix} Y & Y^T [0\ C_{r_c}\ 0]^T + W^T [D_{r_c}\ 0]^T \\ [0\ C_{r_c}\ 0]\, Y + [D_{r_c}\ 0]\, W & \dot{u}^2_{c_{max}} \end{bmatrix} \geq 0, \\
&\begin{bmatrix} 1 & \nu_0^T \\ \nu_0 & Y \end{bmatrix} \geq 0,
\end{aligned}
$$

$$
\left[\begin{array}{cc} Y & Y^T[0\ 0\ C_{r_h}]^T + W^T[0\ D_{r_h}]^T \\ [0\ 0\ C_{r_h}]\,Y + [0\ D_{r_h}]\,W & \dot{u}^2_{h_{max}} \end{array}\right] \geq 0,
$$

$$
\left[\begin{array}{cc} 1 & \nu_0^T \\ \nu_0 & Y \end{array}\right] \geq 0,
$$

$$
\left[\begin{array}{cc} (\sin\phi)(\bar{A}Y + \bar{B}W) + (\sin\phi)(\bar{A}Y + \bar{B}W)^T & -(\cos\phi)(\bar{A}Y + \bar{B}W) + (\cos\phi)(\bar{A}Y + \bar{B}W)^T \\ (\cos\phi)(\bar{A}Y + \bar{B}W) - (\cos\phi)(\bar{A}Y + \bar{B}W)^T & (\sin\phi)(\bar{A}Y + \bar{B}W) + (\sin\phi)(\bar{A}Y + \bar{B}W)^T \end{array}\right.
$$

$$
\left.\begin{array}{c} 0 \\ 0 \\ \bar{A}Y + \bar{B}W + (\bar{A}Y + \bar{B}W)^T + 2\beta Y \end{array}\right] < 0,
$$

$$
\left[\begin{array}{cc} Y & Y^T[0\ C_{a_c}\ 0]^T \\ [0\ C_{a_c}\ 0]\,Y & u^2_{c_{max}} \end{array}\right] \geq 0,
$$

$$
\left.\left[\begin{array}{cc} Y & Y^T[0\ 0\ C_{a_h}]^T \\ [0\ 0\ C_{a_h}]\,Y & u^2_{h_{max}} \end{array}\right] \geq 0.\right\}, \tag{II.73}
$$

where

$$
\bar{A} = \left[\begin{array}{ccc} A & B_cC_{a_c} & B_hC_{a_h} \\ 0 & A_{a_c} & 0 \\ 0 & 0 & A_{a_h} \end{array}\right], \bar{B} = \left[\begin{array}{cc} B_{a_c} & 0 \\ 0 & B_{a_h} \end{array}\right]. \tag{II.74}
$$

Let

$$
\Phi_2(\mathcal{G}_l\,(Z_0, x_{cg_0}\bar{V}_{H_0}, \bar{V}_{C_0}), u_{h_{max}}, \dot{u}_{h_{max}}, u_{c_{max}}, \dot{u}_{c_{max}}, \nu_0) =
$$

$$
\left\{\bar{A}_k, \bar{B}_k, \bar{C}_k, R, S :\right.
$$

$$
\left[\begin{array}{ccc} R & I & \bar{C}_k^T C_{a_c}^T \\ I & S & 0 \\ C_{a_c}\bar{C}_k & 0 & u^2_{c_{max}} \end{array}\right] \geq 0,
$$

$$
\left[\begin{array}{ccc} R & I & \bar{C}_k^T C_{a_e}^T \\ I & S & 0 \\ C_{a_e}\bar{C}_k & 0 & u^2_{h_{max}} \end{array}\right] \geq 0,
$$

$$\left[\begin{array}{ccc} R & I & \bar{C}_k^T B_a^T C_{s_c}^T + R^T [A_a^T C_{r_c}^T \; 0] \\ I & S & A_a^T [C_{r_c}^T \; 0] \\ [0 \; C_{r_c} A_a] \, R + C_{s_c} B_a \bar{C}_k & [0 \; C_{r_c}] \, A_a & \dot{u}_{c_{max}}^2 \end{array}\right] \geq 0,$$

$$\left[\begin{array}{ccc} R & I & \bar{C}_k^T B_a^T C_{s_e}^T + R^T [A_a^T C_{r_e}^T \; 0] \\ I & S & A_a^T [C_{r_e}^T \; 0] \\ [0 \; C_{r_e} A_a] \, R + C_{s_e} B_a \bar{C}_k & [0 \; C_{r_e}] \, A_a & \dot{u}_{h_{max}}^2 \end{array}\right] \geq 0,$$

$$\left[\begin{array}{ccc} 1 & x_0^T & x_0^T S^T \\ x_0 & R & I \\ S x_0 & I & S \end{array}\right] \geq 0,$$

$$\left.\left[\begin{array}{ccc} (\sin\phi)(\Pi^T + \Pi) & (\cos\phi)(-\Pi + \Pi^T) & 0 \\ (\cos\phi)(\Pi - \Pi^T) & (\sin\phi)(\Pi^T + \Pi) & 0 \\ 0 & 0 & \Pi^T + \Pi + \left[\begin{array}{cc} R & I \\ I & S \end{array}\right] 2\beta \end{array}\right] \leq 0, \right\} \tag{II.75}$$

where

$$\Pi = \left[\begin{array}{cc} \underbrace{\left[\begin{array}{ccc} A & B_c C_{a_c} & B_h C_{a_h} \\ 0 & A_{a_c} & 0 \\ 0 & 0 & A_{a_h} \end{array}\right] R + \left[\begin{array}{cc} B_{a_c} & 0 \\ 0 & B_{a_h} \end{array}\right] C_k}_{\bar{A}_k} & \left[\begin{array}{ccc} A & B_c C_{a_c} & B_h C_{a_h} \\ 0 & A_{a_c} & 0 \\ 0 & 0 & A_{a_h} \end{array}\right] \\ S \left[\begin{array}{ccc} A & B_c C_{a_c} & B_h C_{a_h} \\ 0 & A_{a_c} & 0 \\ 0 & 0 & A_{a_h} \end{array}\right] + \bar{B}_k \end{array}\right]. \tag{II.76}$$

Then, a sufficient condition for the existence of a dynamic, output feedbac-controller, or static, state-feedback controller, that stabilizes the feedback system $\mathcal{G}_l\,(Z_0, X_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0})$, does not exceed actuator amplitude and rate limits, and results in acceptable flying qualities in response to the angle-of-attack excursion defined by ν_0, is for the sets, Φ_1, or Φ_2 to be non-empty. At this point, we observed that the same numerical results were obtained using a slightly different approach. Instead of minimzing a parameter $\dot{u}_{max}$ in the LMI decision vector, we exploited the fact that the short period pole becomes more unstable as the center of gravity is moved aft. This allowed the use of a feasibilty algorithm vice minimization algorithm to acheive

the same results. The benefit was a significant reduction in computational time. Now, the *Plant Controller Optimization* (PCO) problem considered in this section can be stated as follows:

Maximize $\{x_{cg}\}$

Subject to:

$$\mathcal{F}(X_0, U_0, x_{cg}, \bar{V}_{H_0}, \bar{V}_{C_0}) = 0,$$

$$\mathcal{H}(X_0, U_0) = Z_0,$$

$$(Y, W) \in \Phi_1(\mathcal{G}_l\ (x_{cg}, \bar{V}_{H_0}, \bar{V}_{C_0}), u_{h_{max}}, \dot{u}_{h_{max}}, u_{c_{max}}, \dot{u}_{c_{max}} \nu_0), \tag{II.77}$$

or

$$(\bar{A}_k, \bar{B}_k, \bar{C}_k, R, S) \in \Phi_2(\mathcal{G}_l\ (x_{cg}, \bar{V}_{H_0}, \bar{V}_{C_0}), u_{h_{max}}, \dot{u}_{h_{max}}, u_{c_{max}}, \dot{u}_{c_{max}} \nu_0). \tag{II.78}$$

A solution to this PCO problem includes a linear controller that stabilizes the feedback system $\mathcal{G}_l\ (Z_0, x_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0})$, and meets actuator limit requirements, as well as a providing a maximum aft center-of-gravity location.

The numerical solution used to map the *Tail Sizing Design Space* involved a binary search over center-of-gravity stations:

1. Fix $\bar{V}_H, \bar{V}_C, V_T, \gamma$. Let $x_{cg_{max}} = 1$ and $x_{cg_{min}} = 0$.
2. $x_{cg_0} = (x_{cg_{max}} + x_{cg_{min}})/2$.
3. Solve $\mathcal{F}(X_0, U_0, x_{cg_0}) = 0$, $\mathcal{H}(X_0, U_0) = Z_0$ for X_0, U_0.
4. Obtain $A(X_0, U_0, x_{cg_0})$, $B(X_0, U_0, x_{cg_0})$ and form $\mathcal{G}_l\ (Z_0, x_{cg_0})$.
5. Solve for $(Y, W) \in \Phi_1(\mathcal{G}_l\ (x_{cg}), u_{h_{max}}, \dot{u}_{h_{max}}, u_{c_{max}}, \dot{u}_{c_{max}}, \nu_0)$ or,
6. Solve for $(\bar{A}_k, \bar{B}_k, \bar{C}_k, R, S) \in \Phi_2(\mathcal{G}_l\ (x_{cg}), u_{h_{max}}, \dot{u}_{h_{max}}, u_{c_{max}}, \dot{u}_{c_{max}}, \nu_0)$.
7. If no such (Y, W) exist
   - $x_{cg_{max}} = x_{cg_0}$,

   else

- $x_{cg_{min}} = x_{cg_0}$.

8. If $x_{cg_{max}} - x_{cg_{min}} > tol$, go to 2.
9. Increment $\bar{V}_H$, go to 1.

4. Results - Rigid Body HSCT

This design methodology was applied to a number of aerodynamic models representative of current high-speed civil transport designs. The results are shown for the aerodynamic model termed Ref A (see Appendix A). Appendix A details how a convenient build-up of the nonlinear longitudinal dynamics in terms of wing-body and tail contributions facilitates the process of mapping the *Tail Sizing Design Space.*

The state feedback synthesis LMIs (set Φ_1) were used. Figure 8 shows an optimization run for a single tail volume of 0.2, no canard, and peak actuator rate limit of 30 degrees per second. As the center-of-gravity location is moved aft, the peak actuator rate approaches the limit. The actuator amplitude remained well below its saturation limit, which was set at 20 degrees of travel from trim.

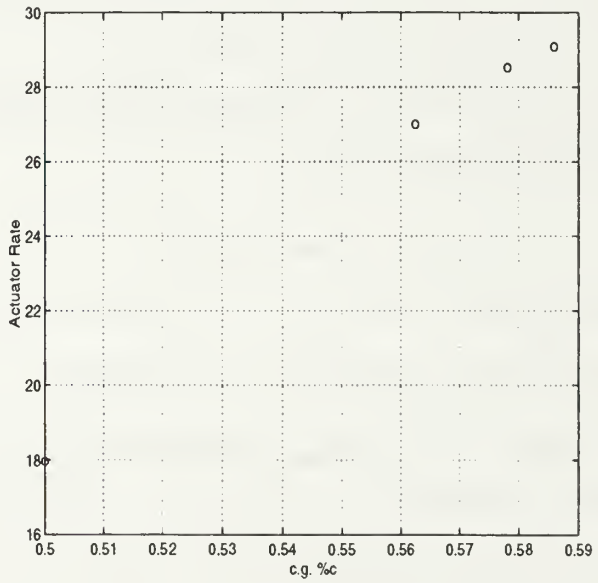


Figure 8. Fixed Tail Volume and Peak Actuator Rate Limit

Figure 9 details the process for one tail volume and a sweep of peak actuator rates. Again, the tail volume was 0.2 and the peak actuator rate limit was incremented

from 5 to 30 degrees per second. Notice, initially the center of gravity moves quickly aft with only small increases in the peak actuator rate required. However, at some point, the peak actuator rate required becomes very sensitive to changes in the center-of-gravity station. For the Ref A model, this break point is in the vicinity of 0.565 percent mean aerodynamic cord.

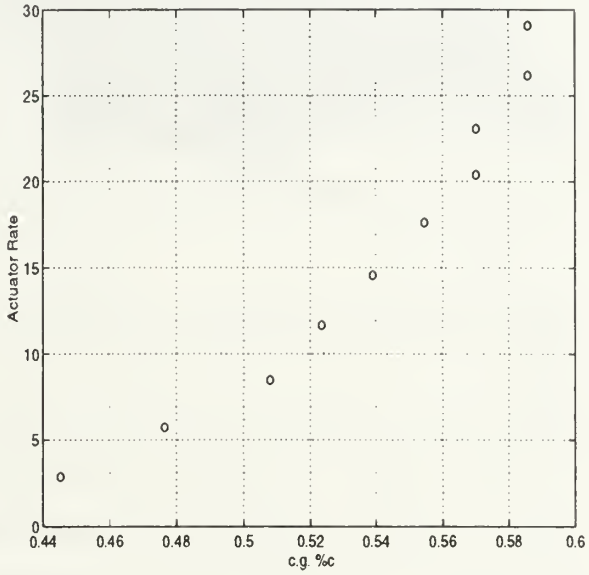


Figure 9. Fixed Tail Volume, Sweep of Peak Actuator Rates

This process is repeated for the range of tail volumes of interest. Figure 10 shows the results of this process for a tail volume of 0.1, and a tail volume of 0.2. A two dimensional representation of the data is cumbersome. When repeated for numerous tail volumes, the *Tail Sizing Design Space* could be viewed in three dimensions. Figure 11 shows the result of fitting a surface to data obtained for a range of tail volumes from 0.1 to 0.3, and a range of peak actuator rates from 5 to 30 degrees per second. The surface represents a lower bound on the peak actuator rate required by the feedback control system to recover from the angle-of-attack excursion, for various combinations of center-of-gravity location and tail volume. An upper bound in the *Tail Sizing Design Space* would be a fixed limit on the peak actuator rate available. Figure 12 shows the inclusion of this plane for a value of 20 degrees per second.

The volume below the plane and above the curved surface represents the *Tail Sizing Design Space*, where linear state-feedback controllers are known to exist that meet design requirements. Figure 13 shows the intersection of the plane representing the peak actuator rate available of 15 degrees per second, with the curved surface representing the minimum peak actuator rate required for a rigid-body HSCT with no canard. This is seen to be an aft center of gravity limit line on the standard center of gravity versus tail volume (scissors) plot.

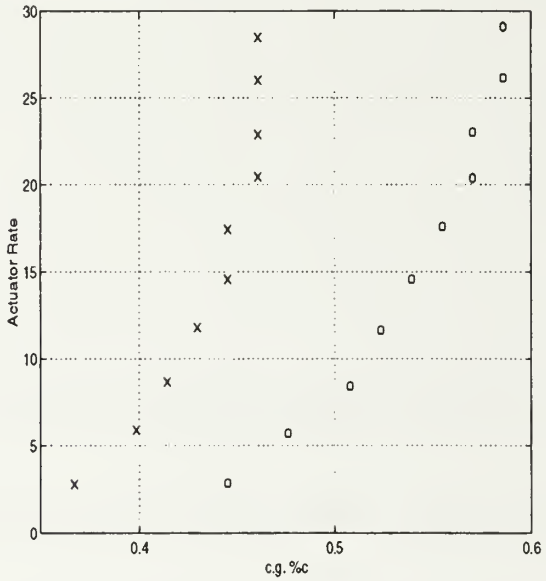


Figure 10. Two Tail Volumes, Sweep of Peak Actuator Rates

At this point, the canard volume was fixed at 0.05. Following the procedure outlined earlier, the lower surface of the *Tail Sizing Design Space* was mapped for a range of horizontal tail volumes from 0.1 to 0.3, and for a range of peak actuator rates from 5 to 40 degrees per second. Amplitude and actuator rate limits for the canard and horizontal tail were matched. Figure 14 shows the resulting lower bound in the design space. Figure 15 shows the results for Ref A HSCT with and without the canard. Figure 16 shows a slice of the two surfaces in Figure 15 at a peak actuator rate limit of 15 degrees per second. This should be familiar as a conventional scissors plot. Of course, as one would expect, the design space increases with the

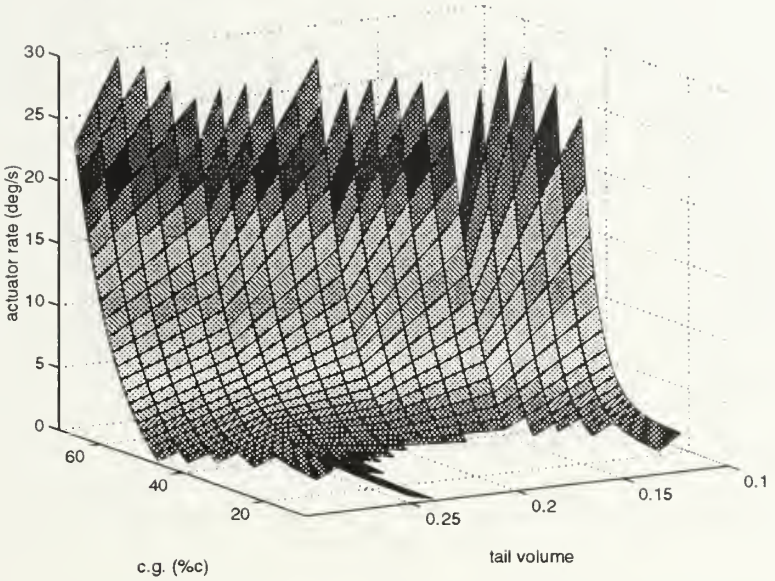


Figure 11. 3D *Tail Sizing Design Space*

additional control power available from the canard. The pertinent question is whether or not there is any benefit in spreading the control power for and aft, so to speak, or if the *Tail Sizing Design Space* would have increased just as much had the tail volume alone been increased by 0.05, and the canard not added. In general, it makes

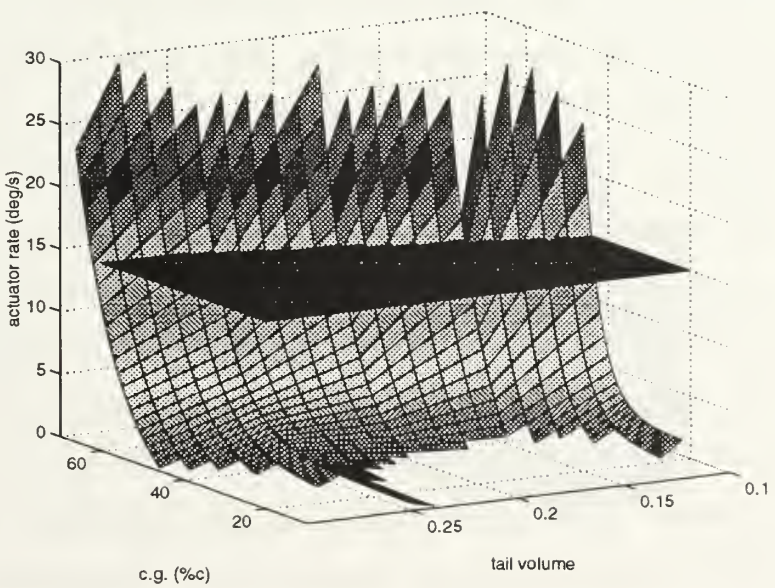


Figure 12. 3D *Tail Sizing Design Space* with Peak Actuator Rate Limit

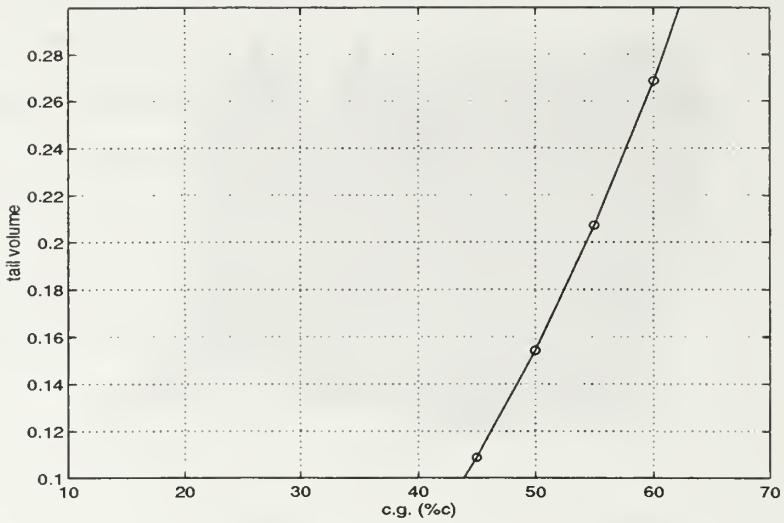


Figure 13. 2D Slice of the *Tail Sizing Design Space*

more sense to compare the competing configurations in terms of equal amounts of combined horizontal tail and canard surface area. For instance, profile drag is more closely related to the surface area of the control surfaces, among other things, and the design goal might be to minimize drag for the same aft center-of-gravity station and actuator rate limit. For this example, which utilized the Ref A data, the distance from the vehicle's wing-body neutral point to the aerodynamic center of the canard or horizontal tail was the same. Therefore, comparisons in terms of normalized area or volume are equivalent.

Figure 17 compares the two configurations, the first without a canard and the second with a canard. The percent change in total control volume required in going from a configuration without a canard to a configuration with a canard is shown as a function of the aft center-of-gravity limit. The same flying quality requirements and actuator limits were used. For the rigid-body HSCT model, the benefit gained from the inclusion of a canard is about a 10 percent savings in total control volume.

This process was repeated, using the output feedback synthesis LMIs (set Φ_2). Figure 18 shows the resulting *Tail Sizing Design Space* obtained utilizing dynamic,

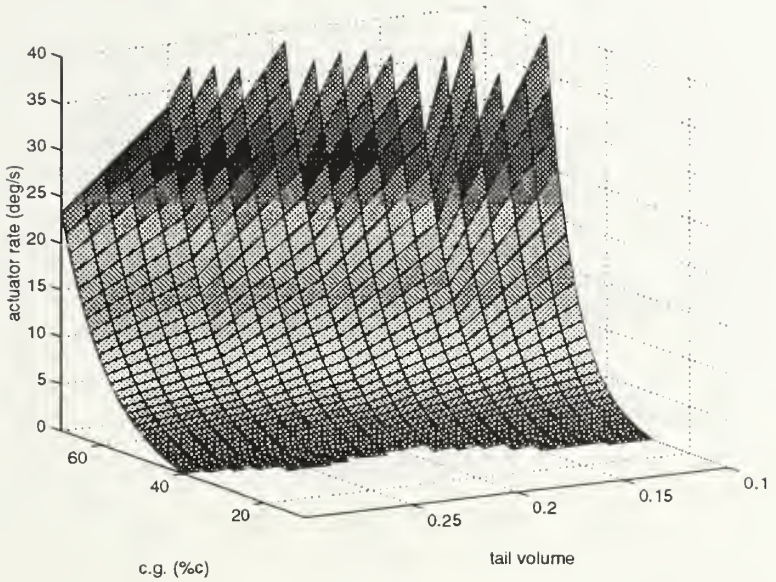


Figure 14. 3D *Tail Sizing Design Space* with Canard

full-state, feedback controllers. Figure 19 compares these results with those obtained utilizing static, state-feedback controllers. The intersection of the two surfaces with the plane representing a target rate limit of 25 degrees per second is shown in Figure 20. By inspection, it is clear that the limits of the *Design Space* are nearly

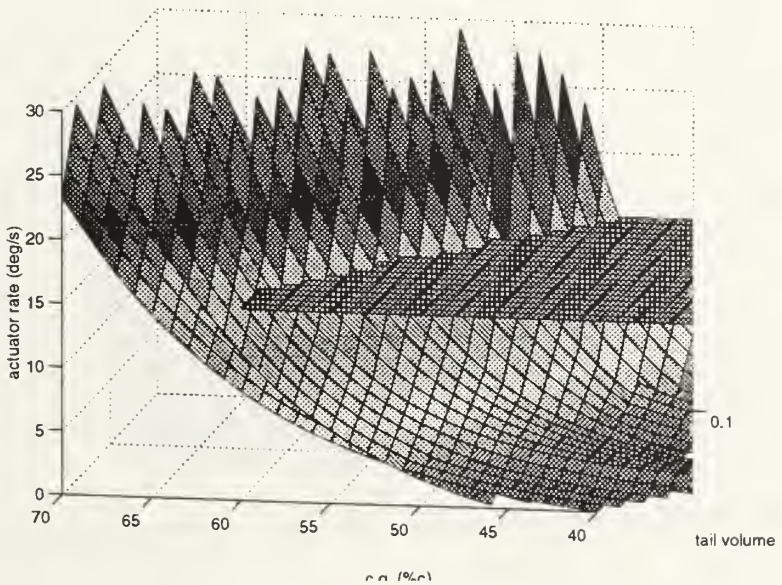


Figure 15. *Tail Sizing Design Space* With and Without a Canard

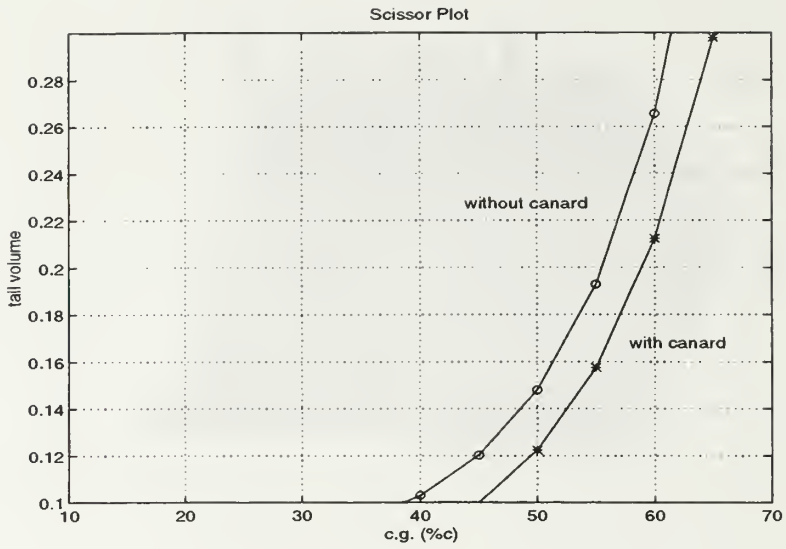


Figure 16. 2D Slice of *Tail Sizing Design Space* With and Without a Canard

identical. This is an important and somewhat surprising result considering the added complexity of the output-feedback controller.

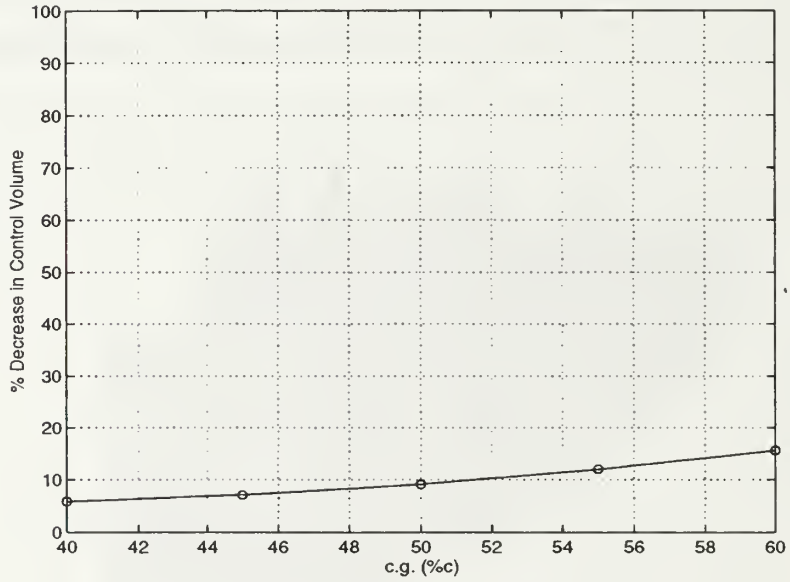


Figure 17. Decrease In Total Control Volume Through the Addition of a Canard

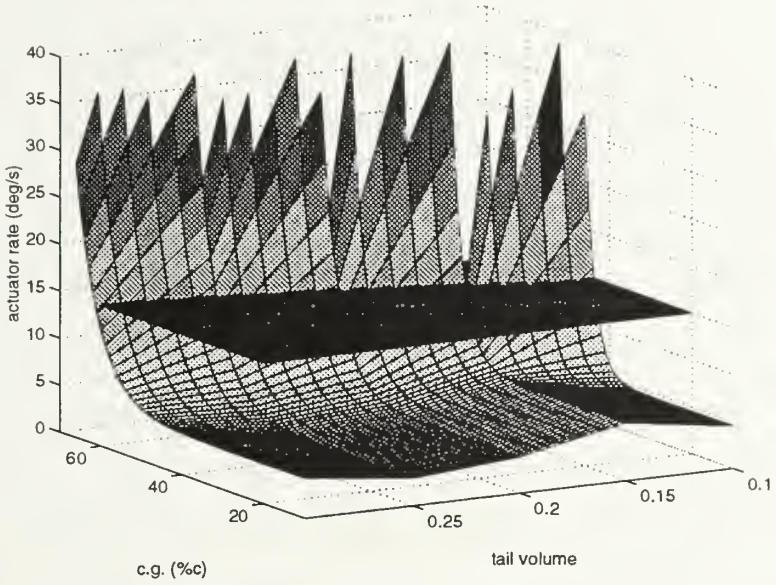


Figure 18. 3D *Tail Sizing Design Space* - Dynamic Controller

5. Problem Formulation - Aeroelastic Model

The development of an integrated aeroelastic aerodynamic model is well documented in work by Waszak and Schmidt [Ref. 45]. The nonlinear equations of motion were derived using a Lagrangian approach and some standard simplifying assump-

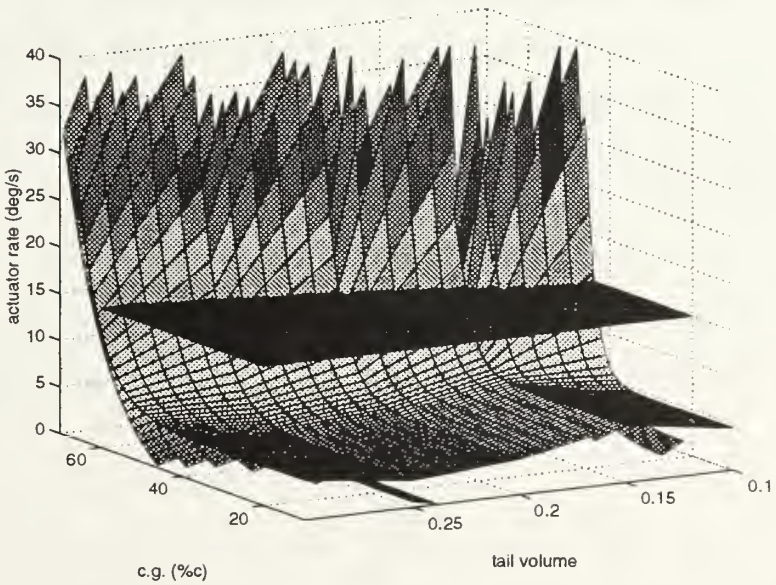


Figure 19. *Tail Sizing Design Space Comparison*: Static versus Dynamic Controllers

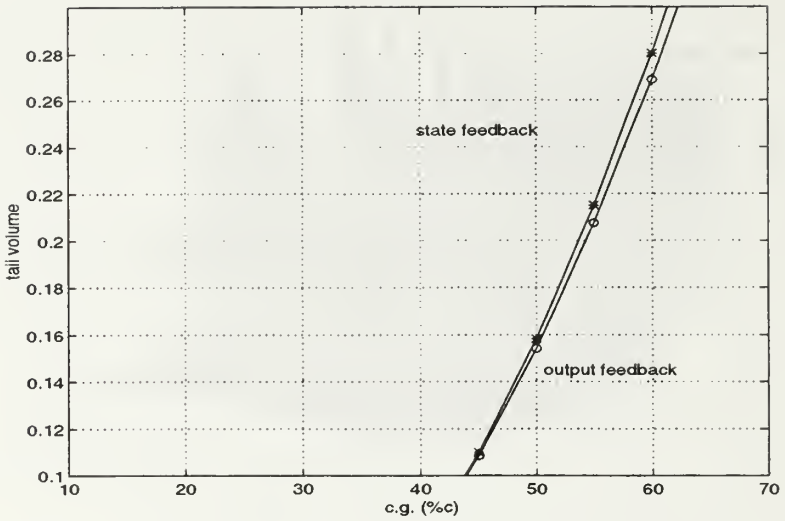


Figure 20. 2D Comparsion: Static versus Dynamic Controller

tions. The elastic deformation is assumed sufficiently small such that linear elastic theory holds. Informally, the flexible deformation is captured by the contribution of an infinite number of states termed the generalized elastic coordinates. Associated with each generalized elastic coordinate is an *in vacuo* mode shape. In laymens terms, this describes the shape of a single elastic mode when fully deflected. Of course, for practical purposes, only a finite number of elastic modes are retained for analysis. In a body-fixed reference coordinate system, the resulting elastic airplane equations of motion for the longitudinal dynamics are:

$$\begin{aligned} m(\dot{u} + qw + gsin\theta) &= Q_x, \\ m(\dot{w} - qu - gcos\theta) &= Q_z, \\ I_{yy}\dot{q} &= Q_\theta, \\ m_i(\ddot{\eta}_i + 2\zeta\omega_n\dot{\eta}_i + \omega_n^2\eta_i) &= Q_{\eta_i}, \end{aligned} \tag{II.79}$$

where,

$$\begin{aligned} Q_i &= \textit{generalized forces}, \\ m_i &= \textit{generalized mass}, \\ \eta_i &= \textit{generalized elastic coordinates}. \end{aligned} \tag{II.80}$$

Let X, and Z be the total aerodynamic and propulsive forces along each of the axis in the body-fixed reference frame. Similarily, let M be the total aerodynamic and propulsive moment about the body-fixed y-axis. Define δx, δz to be virtual displacements along the x and z axis, and define $\delta\theta$ be a virtual rotation about the y axis. When combined with the virtual pertubations of the elastic generalized coordinates, $\delta\eta_i$, the virtual displacements are called the *generalized coordinates.* The Principle of Virtual Work is used to expand the generalized force terms in (II.80) through the use of the generalized coordinates. It states that

$$Q_i = \frac{\partial(\delta W)}{\partial(\delta q_i)}, \tag{II.81}$$

where δW is the work associated with an arbitrary virtual displacement of the generalized coordinates, δq_i. This virtual work done by the aerodynamic and propulsive forces, relative to the inertial reference frame, expressed in a coordinate system attached to the body of the aircraft, is:

$$\delta W = X\delta x + Z\delta z + \left[\bar{M} + (zX - xZ)\right]\delta\theta_B + \int_S \bar{P}(x,z)\cdot\sum_{i=1}^{\infty}\bar{\phi}_i\delta\eta_i dS. \tag{II.82}$$

The last term represents the work done by the distributed surface pressure due to virtual displacements of all of the elastic generalized coordinates.

Applying (II.82) to the expression for virtual work, the generalized forces are seen to be

$$Q_x = \frac{\partial(\delta W)}{\partial(\delta x)} = X, \tag{II.83}$$

$$Q_z = \frac{\partial(\delta W)}{\partial(\delta z)} = Z, \tag{II.84}$$

$$Q_{\theta_B} = \frac{\partial(\delta W)}{\partial(\delta\theta_B)} = \bar{M} + (zX - xZ), \tag{II.85}$$

$$Q_{\eta_i} = \frac{\partial(\delta W)}{\partial(\delta\eta_i)} = \frac{\partial}{\partial(\delta\eta_i)}(\int_S P(x,z)\cdot\bar{\phi}_i\delta\eta_i dS). \tag{II.86}$$

At this point, a method needs to be chosen to determine the aerodynamic and propulsive forces and moments. We assume that a knowledge of wing-body, tail, and canard aerodynamic stability and control derivatives for the rigid-body aircraft are available. Furthermore, the vehicle is assumed to have one flexible mode, with its mode shape and generalized modal mass known. Generally, the first symmetric mode shapes of all the HSCT designs are very similar. Each mode increases the number of states in the linear model by two, which subsequently increases computational times required by the *Tail Sizing Design Tool.* Since the modes retained in the linear model are those that we wish to actively control, the maximum number of modes retained would be three or four. This method can be combined with other methods, [Ref. 45], and used to capture the residuals from the truncated modes. The method used here works well for capturing the interaction of the rigid-body states with the flexible-body states for the first few symmetric modes. It is worth noting that the LMI based *Tail Sizing Design Tool* tool can utilize any method that is capable of generating a linear aeroelastic model at a specified center of gravity, tail volume/canard volume, and flight condition. The main contribution of this method is its suitability to the iterative nature of the numerical solution, and its use of widely available rigid-body aerodynamic stability and control derivatives.

Therefore, the aerodynamic and propulsive forces are expressed in terms of a body-fixed reference frame as

$$\begin{aligned} X &= L\sin\alpha - D\cos\alpha + T_x, \\ Z &= -L\cos\alpha - D\sin\alpha + T_z, \end{aligned} \tag{II.87}$$

where L and D are the lift and drag aerodynamic forces, α is the angle of attack, and T_i is the component of thrust in the i_{th} direction. The total lift of the airplane is

$$L = L_c + L_{wb} + L_t, \tag{II.88}$$

where the subscripts are meant to be suggestive of the canard, wing-body, and tail contributions. In coefficient form, the expression becomes,

$$C_L = \frac{S_c}{S} C_{L_c} + C_{L_{wb}} + \frac{S_t}{S} C_{L_t}, \tag{II.89}$$

where S, S_t, and S_c are the reference area of the wing, tail and canard respectively. For the problem at hand, (II.89) is rewritten explicitly in terms of a tail and canard volume. Let c denote the reference mean aerodynamic cord, $\bar{l}_t$ denote the distance from the aerodynamic center of the tail to the aerodynamic center of the wing-body, and $\bar{l}_c$ be a similar length associated with the canard. Then, (II.89) can be expressed in terms of control volumes as

$$C_L = \bar{V}_C \frac{c}{\bar{l}_c} C_{L_c} + C_{L_{wb}} + \bar{V}_H \frac{c}{\bar{l}_t} C_{L_t}. \tag{II.90}$$

The individual lift coefficients are expanded in a first order Taylor series expansion about the components of the state vector and control inputs. Local changes in the free stream dynamic pressure have been incorporated into the local lift-curve slope derivatives. For example, the coefficient of lift for the tail is expressed as follows:

$$\begin{aligned} C_{L_t} &= C_{L_{t_0}} + C_{L_{t_{\frac{u}{V}}}} \frac{u}{V} + C_{L_{t_\alpha}} \alpha + C_{L_{t_{\dot{\alpha}}}} \dot{\alpha} + C_{L_{t_q}} q \\ &\quad + C_{L_{t_\theta}} \theta + \sum_{i=1}^{\infty} C_{L_{t_{\eta_i}}} \eta_i + \sum_{i=1}^{\infty} C_{L_{t_{\dot{\eta}_i}}} \dot{\eta}_i + C_{L_{t_{u_h}}} u_h. \end{aligned} \tag{II.91}$$

Most of the terms in (II.91) are familiar as aerodynamic stability derivatives for a rigid-body aircraft. Some, such as $C_{L_{t_\eta}}$, may be new to the reader. In order to derive an approximate expression for these aeroelastic stability derivatives, we need

to introduce the concept of a mode shape. It is assumed that the general elastic deformation of the unconstrained body can be described as the product of two functions, one a function of only spatial coordinates, and the other a time dependent function. Informally, the mode shapes, or free vibration modes, are the spatial function, and the generalized elastic coordinates are the time function. Then, the local relative elastic displacement is described in terms of the mode shape, $\bar{\phi}_i(x)$, and the generalized coordinate, $\eta_i(t)$, as

$$\bar{d} = \sum_{i=1}^{\infty} \bar{\phi}_i(x) \cdot \eta_i(t). \tag{II.92}$$

Similarly, the local relative elastic torsion is described as

$$\bar{r} = \sum_{i=1}^{\infty} \frac{d\bar{\phi}_i(x)}{dx} \cdot \eta_i(t). \tag{II.93}$$

Expressions II.92 and II.93 can be used to obtain formulae for the force and moment dependence on flexible motion. Again, considering the tail contribution to lift as an example, the lift-curve slope dependence at the tail on η is computed from the rigid-body aerodynamic stability and control derivatives, and from expressions II.92 and II.93 as

$$C_{L_{t_{\eta_i}}} = C_{L_{t_\alpha}} \left(\frac{d\phi_i(x)}{dx}\right)_{x_{ac_t}}, \tag{II.94}$$

$$C_{L_{t_{\dot{\eta}_i}}} = C_{L_{t_{\dot{\alpha}}}} \left(\frac{\phi_i(x)}{V}\right)_{x_{ac_t}}, \tag{II.95}$$

where x_{ac_t} is the normalized location of the mean aerodynamic center of the tail. The canard and wing-body terms are treated in an analogous fashion. Once the total lift is calculated, an assumed knowledge of a drag polar is used to calculate the total drag.

Similar to the build-up of the total lift on the vehicle, the total pitching moment on the vehicle is expressed in coefficient form. Again, a first order Taylor series expansion about perturbations of the state variables and control inputs results in

$$C_M = C_{M_0} + C_{M_\alpha}\alpha + C_{M_{\dot\alpha}}\dot\alpha + C_{M_q}q + C_{M_{u_h}}u_h + C_{M_{u_c}}u_c + \sum_{i=1}^{\infty} C_{M_{\eta_i}}\eta_i + \sum_{i=1}^{\infty} C_{M_{\dot\eta_i}}\dot\eta_i. \quad \text{(II.96)}$$

The individual derivatives are written in terms of the center-of-gravity location (x_{cg}), aerodynamic center of the wing-body ($x_{ac_{wb}}$), and familiar control volume terms [Ref. 14]. For instance,

$$C_{M_\alpha} = C_{L_\alpha}(x_{cg} - x_{ac_{wb}}) - \bar V_H \frac{\partial C_{L_t}}{\partial \alpha} + \bar V_C \frac{\partial C_{L_c}}{\partial \alpha} + \frac{\partial C_{M_p}}{\partial \alpha}, \quad \text{(II.97)}$$

and

$$C_{M_{\eta_i}} = C_{L_{\eta_i}}(x_{cg} - x_{ac_{wb}}) - \bar V_H \frac{\partial C_{L_t}}{\partial \eta_i} + \bar V_C \frac{\partial C_{L_c}}{\partial \eta_i} + \frac{\partial C_{M_p}}{\partial \eta_i}, \quad \text{(II.98)}$$

which are composed of stability and control derivatives that were either assumed provided or previously computed.

The remaining term to consider involves expansion of the generalized force associated with the elastic generalized coordinates. Recall,

$$Q_{\eta_i} = \frac{\partial}{\partial(\delta\eta_i)}\left(\int_S P(x,z)\cdot\bar\phi_i\delta\eta_i dS\right). \quad \text{(II.99)}$$

As a first step, the integral expression in (II.99) is separated into three parts:

$$\int_S P(x,z)\cdot\bar\phi_i\delta\eta_i dS = \int_c P(x,z)\cdot\bar\phi_i\delta\eta_i dS + \int_{wb} P(x,z)\cdot\bar\phi_i\delta\eta_i dS + \int_t P(x,z)\cdot\bar\phi_i\delta\eta_i dS. \quad \text{(II.100)}$$

The pressure distribution over each surface is approximated by a point force acting at the aerodynamic center of the lifting surfaces of the aircraft, and the axial component is ignored. Therefore, the three integrals above are approximated as

$$\int_S P(x,z)\cdot\bar{\phi}_i\delta\eta_i dS \approx F_{Z_c}\bar{\phi}_i(x_{ac_c})\delta\eta_i + F_{Z_{wb}}\bar{\phi}_i(x_{ac_{wb}})\delta\eta_i + F_{Z_t}\bar{\phi}_i(x_{ac_t})\delta\eta_i. \tag{II.101}$$

Then, using (II.99), a reasonable approximation of the generalized forces is

$$Q_{\eta_i} \approx F_{Z_c}\bar{\phi}_i(x_{ac_c}) + F_{Z_{wb}}\bar{\phi}_i(x_{ac_{wb}}) + F_{Z_t}\bar{\phi}_i(x_{ac_t}). \tag{II.102}$$

Each term in (II.102) is expanded in a first order Taylor series expansion. The force on the tail, for instance, becomes

$$\begin{aligned} F_{Z_t} = {} & F_{Z_{t_{\frac{u}{V}}}}\frac{u}{V} + F_{Z_{t_\alpha}}\alpha + F_{Z_{t_{\dot{\alpha}}}}\dot{\alpha} + F_{Z_{t_q}}q + F_{Z_{t_\theta}}\theta \\ & + \sum_{i=1}^{\infty} F_{Z_{t_{\eta_i}}}\eta_i + \sum_{i=1}^{\infty} F_{Z_{t_{\dot{\eta}_i}}}\dot{\eta}_i + F_{Z_{t_{u_h}}}u_h. \end{aligned} \tag{II.103}$$

Recall, the generalized forces, Q_η, drive the dynamics of the elastic generalized coordinates. The rigid-body states couple into the elastic states via terms like

$$F_{Z_{t_\alpha}}\bar{\phi}_i\alpha(x_{ac_t}) = \cos\alpha q S\bar{V}_H\frac{c}{\bar{l}_t}C_{L_{t_\alpha}}\bar{\phi}_i(x_{ac_t})\alpha, \tag{II.104}$$

and the control inputs couple into the elastic states through terms like

$$F_{Z_{t_{u_h}}}\bar{\phi}_i(x_{ac_t})u_h = \cos\alpha q S\bar{V}_H\frac{c}{\bar{l}_t}C_{L_{t_{u_h}}}\bar{\phi}_i(x_{ac_t})u_h. \tag{II.105}$$

The remaining terms couple the elastic states among themselves. Using the fifth term in (II.103) for an example, we obtain

$$F_{Z_{t_{\eta_i}}} = \cos\alpha q S \bar{V}_H \frac{c}{\bar{l}_t} C_{L_{t_\alpha}} (\frac{d\bar{\phi}_i(x)}{dx})_{x_{ac_t}} \bar{\phi}_i(x_{ac_t}), \tag{II.106}$$

and

$$F_{Z_{t_{\dot{\eta}_i}}} = \cos\alpha q S \bar{V}_H \frac{c}{\bar{l}_t} C_{L_{t_\alpha}} \bar{\phi}_i(x_{ac_t}) \frac{\bar{\phi}_i(x_{ac_t})}{V}. \tag{II.107}$$

This completes the description of the modeling of the nonlinear aeroelastic dynamics. The intent was not to document each term. That level of detail is left to Appendix A. Instead, the flavor of how to incorporate the influence of a few elastic modes was demonstrated. The method assumed a knowledge of wing-body, tail and canard aerodynamic stability and control derivatives for the rigid-body, and a set of mode shapes and generalized modal masses.

The nonlinear aeroelastic equations of motion were linearized at an equilibrium point determined by the flight condition, canard volume (possibly zero), tail volume, and center of gravity. The resulting linear model is given by:

$$\begin{bmatrix} \dot{x}_R \\ \dot{x}_E \end{bmatrix} = \begin{bmatrix} A_{RR} & A_{RE} \\ A_{ER} & A_{EE} \end{bmatrix} \begin{bmatrix} \dot{x}_R \\ \dot{x}_E \end{bmatrix} + \begin{bmatrix} B_R \\ B_E \end{bmatrix} \begin{bmatrix} u_c \\ u_h \end{bmatrix}, \tag{II.108}$$

where

$$\begin{aligned} x_R &= \begin{bmatrix} u & w & q & \theta \end{bmatrix}^T, \\ x_E &= \begin{bmatrix} \eta & \dot{\eta} \end{bmatrix}^T. \end{aligned}$$

Generalized elastic coordinates are somewhat lacking in physical intuition, and are not directly measured by any sensor suite. The relationships between sensed pitch angle and pitch rate at a given local body station (subscript l), the rigid-body states (subscript R), and the generalized elastic coordinates are:

$$\theta_l = \theta_R - \sum_{i=1}^{\infty} (\frac{d\bar{\phi}_i}{dx})_{x_l} \eta_i, \tag{II.109}$$

$$q_l = \dot{\theta}_l, \tag{II.110}$$

$$= \dot{\theta}_R - \sum_{i=1}^{\infty} (\frac{d\bar{\phi}_i}{dx})_{x_l} \dot{\eta}_i. \tag{II.111}$$

Let

$$T = \begin{bmatrix} & I_{4x4} & & & 0_{4x2} & \\ 0 & 0 & 1 & 0 & 0 & (\frac{d\bar{\phi}}{dx})_{x_l} \\ 0 & 0 & 0 & 1 & (\frac{d\bar{\phi}}{dx})_{x_l} & 0 \end{bmatrix} \tag{II.112}$$

define a similarity transformation that replaces the generalized elastic coordinates, η, $\dot{\eta}$, with a more physically meaningful pair of states, such as q_l and θ_l. The location chosen for the placement of the local pitch rate and pitch angle sensors was the cockpit station. Typically, an area is sought where the mode shape slope has its most positive value [Ref. 1]. The linearized aeroelastic model now becomes

$$\begin{bmatrix} \dot{x}_R \\ \begin{bmatrix} \dot{q}_l \\ \dot{\theta}_l \end{bmatrix} \end{bmatrix} = T \begin{bmatrix} A_{RR} & A_{RE} \\ A_{ER} & A_{EE} \end{bmatrix} T^{-1} \begin{bmatrix} x_R \\ \begin{bmatrix} q_l \\ \theta_l \end{bmatrix} \end{bmatrix} + T \begin{bmatrix} B_R \\ B_E \end{bmatrix} \begin{bmatrix} u_c \\ u_h \end{bmatrix},$$

$$=: A_f \begin{bmatrix} x_R \\ \begin{bmatrix} q_l \\ \theta_l \end{bmatrix} \end{bmatrix} + B_f \begin{bmatrix} u_c \\ u_h \end{bmatrix}. \tag{II.113}$$

Using Ref A data from Appendix A, the behavior of a rigid-body HSCT aircraft is compared with an aeroelastic model with the first elastic mode retained. Table I compares the eigenvalues of the two models. Figure 21 highlights the undesirable effect of the elastic motion. It shows the pitch rate sensed at the cockpit to a step input of the elevator for the two models.

The frequency separation between the elastic mode and the short period dynamics is approximately 1 Hertz, and the damping of the flexible mode is only 0.02. Typically, attempts are made to attenuate the feedback prior to excitation of the flexible dynamics. On large transport aircraft with the flexible dynamics close in

	Short Period		Long Period		Flexible Mode	
	frequency	damping	frequency	damping	frequency	damping
Aeroelastic	0.90	.75	0.14	0.55	6.7	0.02
Rigid-body	0.91	.75	0.14	0.07	n/a	n/a

Table I. Eigenvalues of Aeroelastic and Rigid-body Models

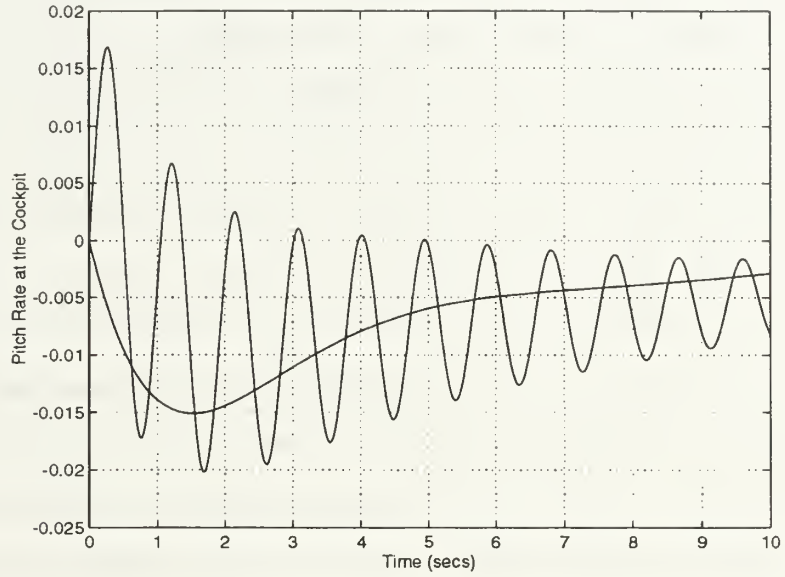


Figure 21. Pitch Rate at Cockpit, Aeroelastic vs. Rigid Body

frequency to the short period dynamics, this is hardly possible. Furthermore, even if suitable notch or low-pass filtering within the control loop could be attained, the extremely light damping of the flexible modes results in problems in terms of gust-induced structural responses and fatigue life. Therefore, the problem posed will be one of actively controlling the flexible modes retained. Generally, this will entail improving the damping of the flexible dynamics, while ensuring stability of the short period dynamics as the center of gravity is moved aft.

As before, actuator dynamics are appended to the aeroelastic model. The Plant-Controller Optimization problem formulation is exactly the same as discussed in section 1.

6. Results - Aeroelastic Model

The results are shown for the aeroelastic dynamic model termed Ref B in Appendix A. Ref B utilizes the same rigid-body stability and control derivatives as Ref A. The effect of a single, symmetric, flexible mode was incorporated. The mode shape is shown in Figure 22. It was experimentally determined that requiring the feedback controller to increase the damping of the flexible mode to 0.05 resulted in acceptable damping of the short period mode. Once again, the LMI based *Tail Sizing Design Tool* was used to map out a surface in the *Tail Sizing Design Space.* Figure 23 shows the design space for Ref B without a canard. An upper bound in the *Tail Sizing Design Space* is shown by the level plane at a peak actuator rate of 25 degrees per second. Figure 24 compares the aeroelastic model to the rigid-body model. Obviously, considerably more actuator rate is required for the aeroelastic model in order to recover from the angle-of-attack excursion, while actively controlling the flexible dynamics. Intuitively, this seems obvious. The *Tail Sizing Design Tool* provides a metric for comparison. For example, consider a target tail volume of 0.20 and center-of-gravity station of 45 percent mean aerodynamic cord. This is well within the *Tail Sizing Design Space* of the rigid-body model, but outside that of the aeroelastic model. The aeroelastic model requires either an extra 0.05 increase in tail volume, or 14 degrees per second increase in peak actuator rate, to move within its *Tail Sizing Design Space.* In some respects, this is the price to be paid for actively controlling a non-rigid vehicle.

Next, the effect of the addition of a canard to the aeroelastic model is explored. As before, a canard volume of 0.05 was selected, and the design space was determined. The rest of the design requirements remained unchanged, and the results are shown in Figure 25. Figure 26 compares the design space of the aeroelastic model with and without the canard. Figure 27 shows the resulting aft line on the tail volume sizing plot when the two are cut at a peak actuator rate of 25 degrees per second. Of course, the added control volume due to the canard allows a further aft center-of-

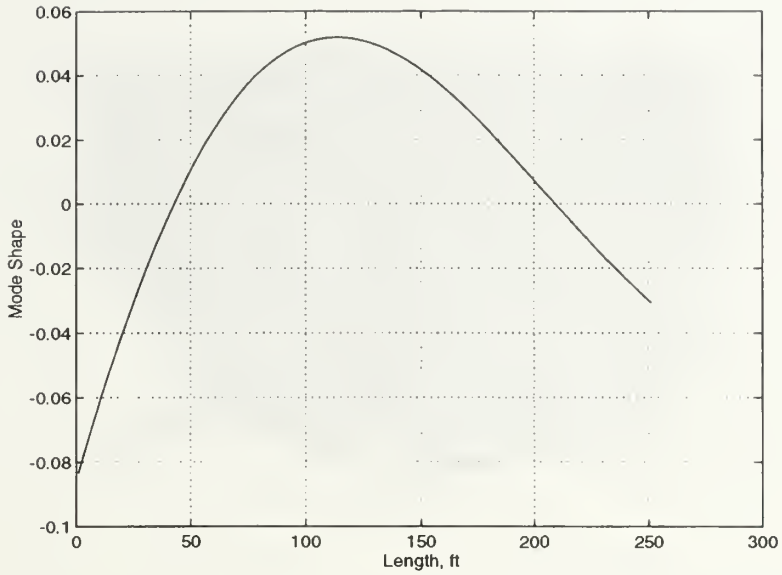


Figure 22. 1^{st} Symmetric Mode Shape

gravity location for a similar actuator rate limit.

The relative effect of adding a canard is addressed by comparing the total control volume, $(\bar{V}_H + \bar{V}_C)$, required for a given aft center-of-gravity limit and peak actuator rate limit. The results are shown in Figure 28 as a percent reduction in

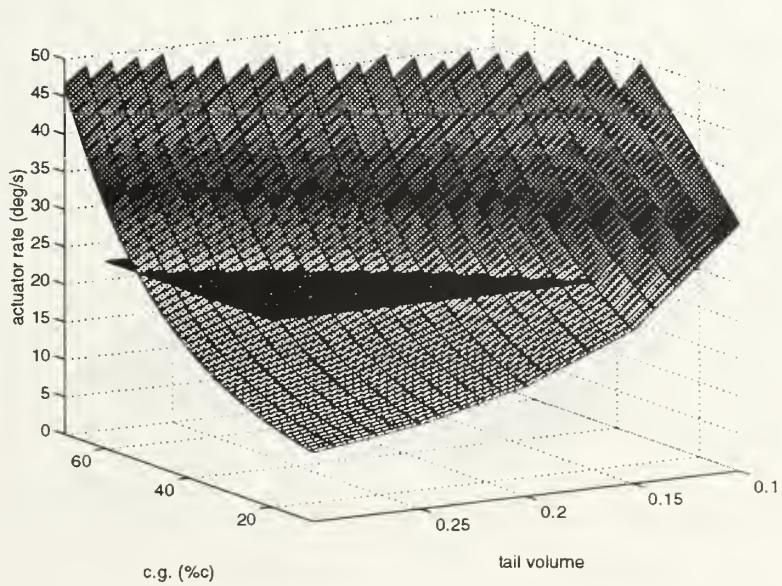


Figure 23. 3D *Tail Sizing Design Space*: Aeroelastic Model

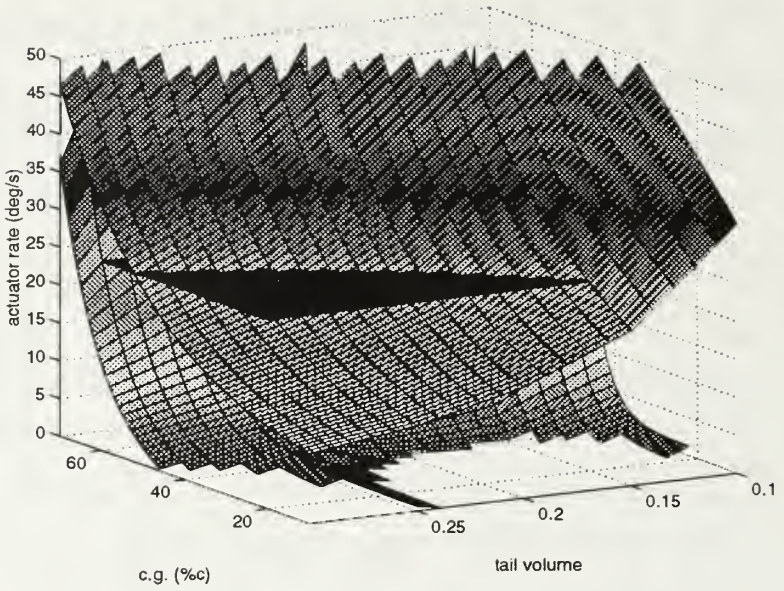


Figure 24. *Tail Sizing Design Space*: Aeroelastic versus Rigid Body Model

control volume required in going from the configuration without a canard to the configuration with a canard. Since the lengths of the tail and canard moment arms are the same, Figure 28 could also represent a savings in total control effector surface area. Experience has shown that the use of canards is desirable for flexible aircraft

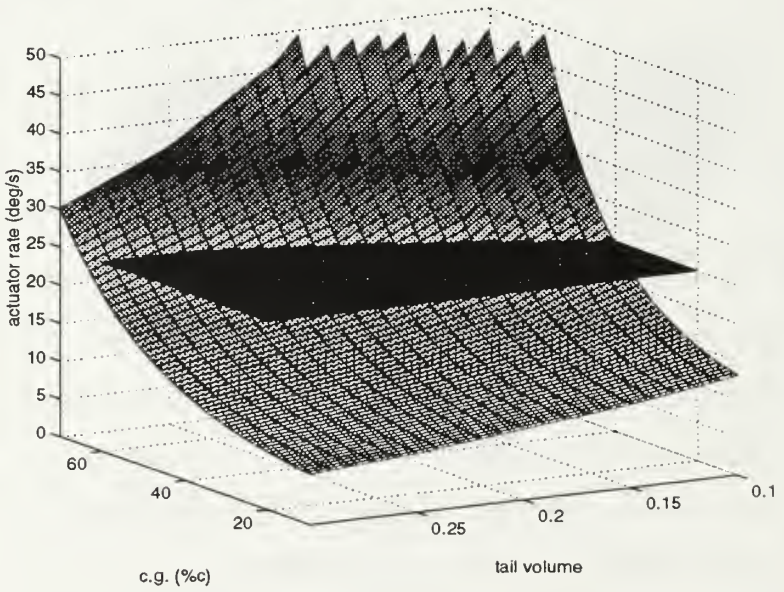


Figure 25. *Tail Sizing Design Space* With Canard

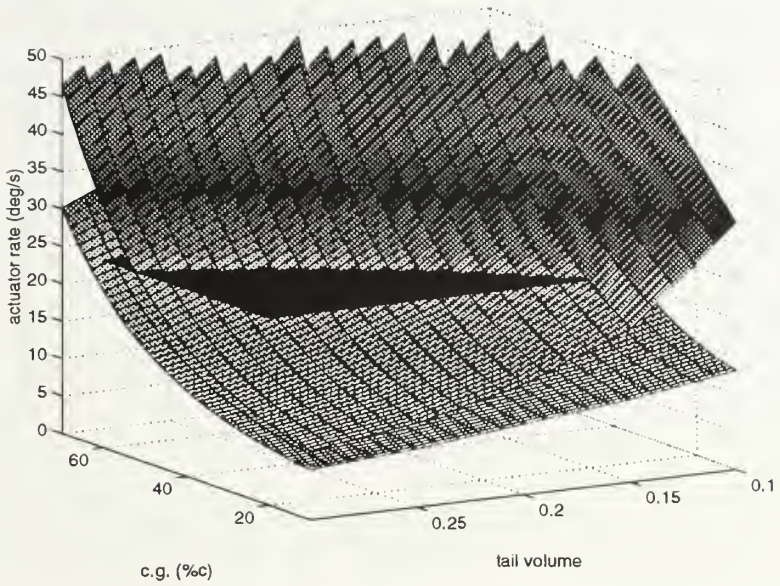


Figure 26. *Tail Sizing Design Space Comparison*: Canard On and Off

[Ref. 1]. This method provides a metric to quantify that benefit. In this example, the inclusion of a canard is 300 percent more effective when added to the aeroelastic model then when added to the rigid-body model.

As before, the process was repeated using the output feedback LMIs. The

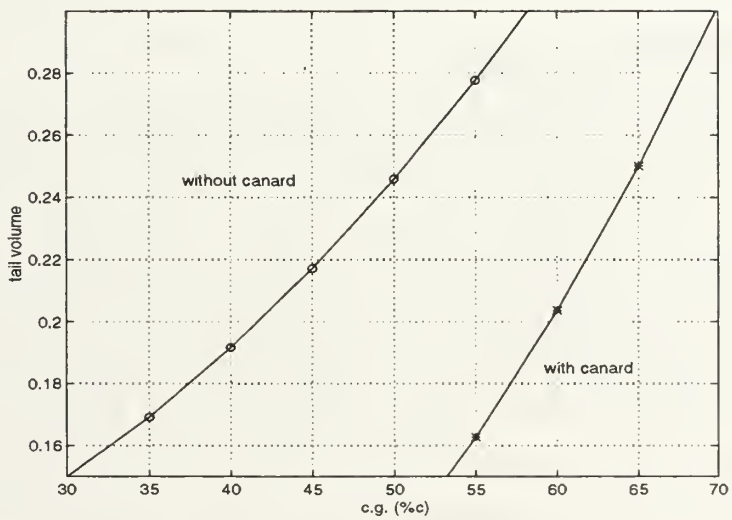


Figure 27. 2D Comparison: Canard On and Off

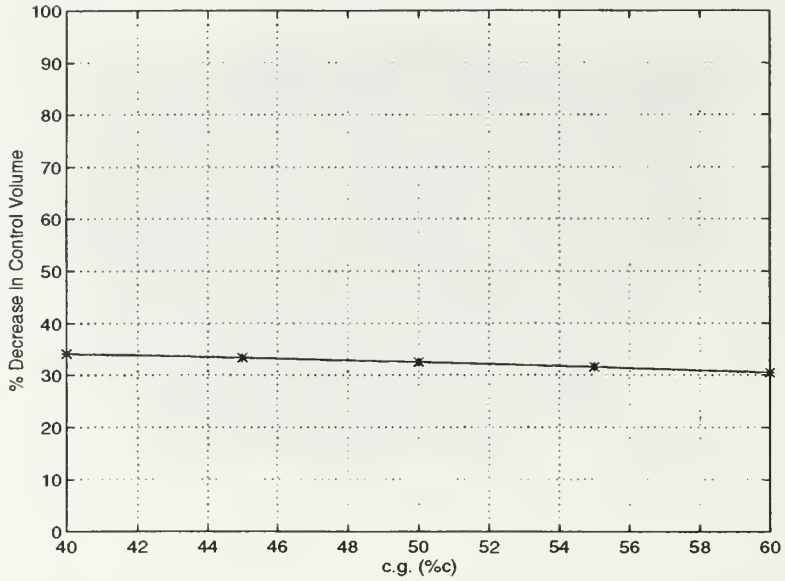


Figure 28. Decrease In Total Control Volume Through the Addition of a Canard

results utilizing the aeroelastic model support the conjecture that dynamic controllers of the same order as the plant do no better than static, state-feedback controllers. A representative comparison is shown in Figure 29. There the aft center-of-gravity stations are shown for a range of tail volumes and fixed canard volume. The peak actuator rate limit was 40 degrees per second.

D. GUST RECOVERY

A second dynamic requirement imposed on transport aircraft is the ability to recover from a severe gust. The rationale is that the open loop HSCT should not be so unstable that an unrealistically fast actuator and flight control system are required. The gust recovery criterion is similar to that of the high angle-of-attack excursion in many respects. Both are at their most adverse when the vehicle is at a slow speed. Also, the aft center-of-gravity configuration is limiting. However, unlike the high angle-of-attack excursion, which is assumed to occur essentially instantly, gust recovery criterion involves the vehicle penetrating a particular wind-shear over time.

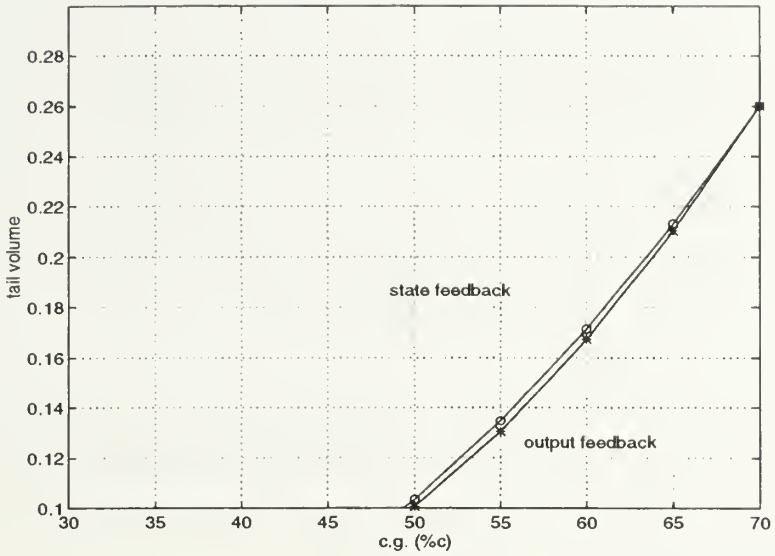


Figure 29. 2D *Tail Sizing Design Space* Comparison: Static versus Dynamic Controller on an Aeroelastic Model

The wind-shear occurs in the vertical plane, begins at zero velocity, builds to some peak value, and then dies out again. Since the shear is defined to occur as the vehicle crosses into a fixed region in space, the rate at which it is penetrated depends on the flight speed of the vehicle. In some industry documents, the gust profile is defined to reach its peak within 30 feet of travel of the vehicle. However, there is ambiguity as to the rate and peak value of the gust. The bottom line is that relatively late in the design/testing process, a flight simulation will be conducted. The vehicle will penetrate some gust, based on empirical weather data gathered from airports throughout the country. Failure at this stage involves a major redesign. What is required is the ability to quantify the concepts used in the definition of the design requirement, such as *so unstable*, and *unrealistically fast actuator*. The *Tail Sizing Design Tool* provides a bench mark to assess competing configurations that is directly related to recovery from a gust penetration.

1. Additional LMI Considerations

Recall the state feedback synthesis LTI system description.

$$\left\{\begin{array}{lll} \dot{x} &= Ax + Bu & \\ z &= Cx + Du &= \left\{\begin{array}{ll} \dot{x} &= [A + BK]x \\ z &= [C + DK]x, \end{array}\right. \\ u &= Kx & \end{array}\right. \tag{II.114}$$

and suppose that the desired block structure of the feedback gain, K, is

$$K = \left[\begin{array}{cc} K_1 & 0 \end{array}\right]. \tag{II.115}$$

A sufficient condition to assure the proper block structure of the feedback gain is to specify an appropriate block structure to the unknown variables in the corresponding LMIs. Recall, for the state synthesis feedback problem, the feedback gain is recovered as $K = WY^{-1}$. Therefore, specify

$$Y = \left[\begin{array}{cc} Y_1 & 0 \\ 0 & Y_2 \end{array}\right], \tag{II.116}$$

$$W = \left[\begin{array}{cc} W_1 & 0 \\ W_2 & 0 \end{array}\right], \tag{II.117}$$

such that

$$\begin{aligned} K &= \left[\begin{array}{cc} W_1 & 0 \\ W_2 & 0 \end{array}\right]\left[\begin{array}{cc} Y_1^{-1} & 0 \\ 0 & Y_2^{-1} \end{array}\right], \\ &= \left[\begin{array}{cc} W_1 Y_1^{-1} & 0 \\ W_2 Y_1^{-1} & 0 \end{array}\right]. \end{aligned}$$

2. Problem Formulation

The gust recovery requirement describes the face of the gust profile as a *One Minus Cosine* disturbance. An acceptable alternative to the *One Minus Cosine* profile is to fit two exponential functions to the profile. This captures the important rapid change in airmass velocity as the vehicle penetrates the shear. Unlike the *One Minus Cosine* disturbance, the gust velocity returns to zero after reaching its peak. The rate of decay of the gust velocity after reaching its peak, however, can be made quite slow. Therefore, the impact on the vehicle's dynamics is minimal.

Let t_p be the time it takes the HSCT to penetrate the shear, and let G_P be the peak vertical velocity of the shear. Then, the following functions define the two wind-shear profiles.

One Minus Cosine Profile:

$$\begin{aligned} f_1(t) &= \frac{G_p}{2}(1-\cos(\frac{t\,\pi}{t_p})), \qquad 0<t<t_p \\ &= G_p, \qquad\qquad\qquad\quad t \geq t_p. \end{aligned} \tag{II.118}$$

Double Decaying Exponential Profile:

$$f_2(t) = -G_p \exp^{(-\lambda_1 t)} + G_p \exp^{(-\lambda_2 t)}, \qquad 0<t, \tag{II.119}$$

where

$$\lambda_1 = 3t_p, \quad \text{and} \quad \lambda_2 = \frac{\lambda_1}{100}. \tag{II.120}$$

Both wind-shear profiles are shown in Figure 30.

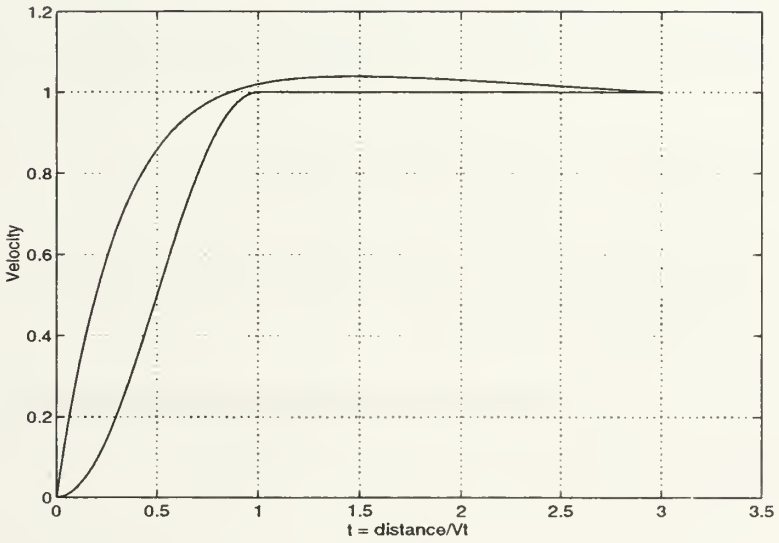


Figure 30. Two Wind shear Profiles

Let the vehicle's linearized longitudinal dynamics be given by the LTI system

$$
\begin{aligned}
\dot{\nu} &= \bar{A}\nu + \bar{B}u, \\
&= \left[\left[\begin{array}{cccc} A_u & A_w & A_q & A_\theta \end{array} \right] \quad A_{act} \right] \nu + \bar{B}u, \\
\nu &= \left[\begin{array}{cc} x & x_a \end{array} \right],
\end{aligned}
\tag{II.121}
$$

as before. Based on (II.119), the state space description of the wind-shear is

$$
\begin{aligned}
\dot{x}_g &= \left[\begin{array}{cc} -\lambda_1 & 0 \\ 0 & -\lambda_2 \end{array} \right] x_g, \\
&= A_g x_g \\
v_g &= \left[\begin{array}{cc} 1 & 1 \end{array} \right] x_g, \\
x_{g_0} &= [-G_p \; G_p]^T,
\end{aligned}
\tag{II.122}
$$

where v_g is the vertical velocity of the shear. Now, a model of the vehicle's dynamics in an airmass whose vertical velocity is $v_g(t)$ is

$$
\begin{aligned}
\dot{\eta} &= \left[\begin{array}{cc} \bar{A} & \left[\begin{array}{cc} (A_u \sin\theta_0 + A_w \cos\theta_0) & (A_u \sin\theta_0 + A_w \cos\theta_0) \end{array} \right] \\ 0 & A_g \end{array} \right] \eta + \left[\begin{array}{c} \bar{B} \\ 0 \end{array} \right] u, \\
&=: A_I \eta + B_I u, \\
\eta &= \left[\begin{array}{cc} \nu & x_g \end{array} \right], \\
\eta_0^T &= \left[\begin{array}{ccc} 0 & -G_p & G_p \end{array} \right]^T.
\end{aligned}
$$

The velocity states in (II.123) are inertially referenced quantities. Consider a similarity transformation based on the change of variables,

$$\zeta = T\eta,$$

where

$$T = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{V_t} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} -\sin\theta_0 & -\sin\theta_0 \\ \frac{-\cos\theta_0}{V_t} & \frac{-\cos\theta_0}{V_t} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \\ 0 & I \end{bmatrix},$$

and

$$\begin{aligned} \dot{\zeta} &= TA_IT^{-1}\zeta + TB_Iu, \\ &=: \tilde{A}\zeta + \tilde{B}u, \\ \zeta_0 &= T\eta_0. \end{aligned} \tag{II.123}$$

Then, the first two states of ζ are perturbations in true airspeed and angle of attack, both airmass relative quantities. This makes the first four states of ζ natural quantities to be considered for feedback. However, for all practical purposes, the gust states are not measured, and should not be considered for feedback. This implies that the feedback considered should be of the form:

$$K = \begin{bmatrix} K_1 & 0 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta 2 \end{bmatrix}, \tag{II.124}$$

where ζ_2 corresponds to the gust states, and ζ_1 corresponds to all of the remaining states.

3. Proposed Numerical Solution

Let

$$\Phi_3(\mathcal{G}_l\,(Z_0, x_{cg_0}, \bar{V}_{H_0}, \bar{V}_{C_0}), u_{h_{max}}, \dot{u}_{h_{max}}, \zeta_0) =$$
$$\left\{ W, Y > 0 : \qquad Y = \begin{bmatrix} Y_1 & 0 \\ 0 & Y_2 \end{bmatrix}, \quad W = \begin{bmatrix} W_1 & 0 \\ W_2 & 0 \end{bmatrix}, \right.$$

$$
\left[\begin{array}{cc} 1 & \zeta_0^T \\ \zeta_0 & Y \end{array}\right] \geq 0,
$$

$$
\left[\begin{array}{cc} Y & Y^T [0\ 0\ C_{r_h}\ 0\ 0]^T + W^T [0\ D_{r_h}]^T \\ [0\ 0\ C_{r_h}\ 0\ 0]\, Y + [0\ D_{r_h}]\, W & \dot{u}^2_{h_{max}} \end{array}\right] \geq 0,
$$

$$
\left[\begin{array}{cc} 1 & \zeta_0^T \\ \zeta_0 & Y \end{array}\right] \geq 0,
$$

$$
\left[\begin{array}{cc} (\sin\phi)(\tilde{A}Y + \tilde{B}W) + (\sin\phi)(\tilde{A}Y + \tilde{B}W)^T & -(\cos\phi)(\tilde{A}Y + \tilde{B}W) + (\cos\phi)(\tilde{A}Y + \tilde{B}W)^T \\ (\cos\phi)(\tilde{A}Y + \tilde{B}W) - (\cos\phi)(\tilde{A}Y + \tilde{B}W)^T & (\sin\phi)(\tilde{A}Y + \tilde{B}W) + (\sin\phi)(\tilde{A}Y + \tilde{B}W)^T \\ 0 & 0 \\ 0 & 0 \\ \tilde{A}Y + \tilde{B}W + (\tilde{A}Y + \tilde{B}W)^T + 2\beta Y & \end{array}\right] < 0,
$$

$$
\left.\left[\begin{array}{cc} Y & Y^T [0\ 0\ C_{a_h}\ 0\ 0]^T \\ [0\ 0\ C_{a_h}\ 0\ 0]\, Y & u^2_{h_{max}} \end{array}\right] \geq 0. \right\}. \tag{II.125}
$$

Now, replace Φ_1 with Φ_3 in (II.77), and proceed with the algorithm outlined in section (3).

4. Gust Recovery Results

This design tool was being developed concurrently with a NASA led multi-corporate effort to define a baseline configuration for an HSCT vehicle. The design engineers desired the ability to quickly asses the influence of changes in key parameters of the wind-shear on the final design. This necessitated that computational time be kept short, on the order an hour or less, since the intention was to investigate multiple scenarios throughout the day.

In that light, the gust recovery criterion was applied to the numerical model, Ref A, with a fixed tail volume of 0.11 and no canard. Still to be finalized were the design requirements concerning the definition of the wind-shear. The design team desired to know how sensitive their designs were to changes in peak values of the wind-shear and/or time to penetrate the wind-shear. The *Tail Sizing Design Tool* was used to compute aft center-of-gravity stations for differing wind shear profiles.

Figure 31 shows the sensitivity of the aft center of gravity location to changes in the peak value of the gust. The penetration distance is one cord length. The peak

gust velocities were 30, 45, and 60 feet per second. The data indicates that at around 20 degrees per second peak actuator rate, every 50 percent increase in the peak gust velocity pushes the aft center-of-gravity station forward 10 percent.

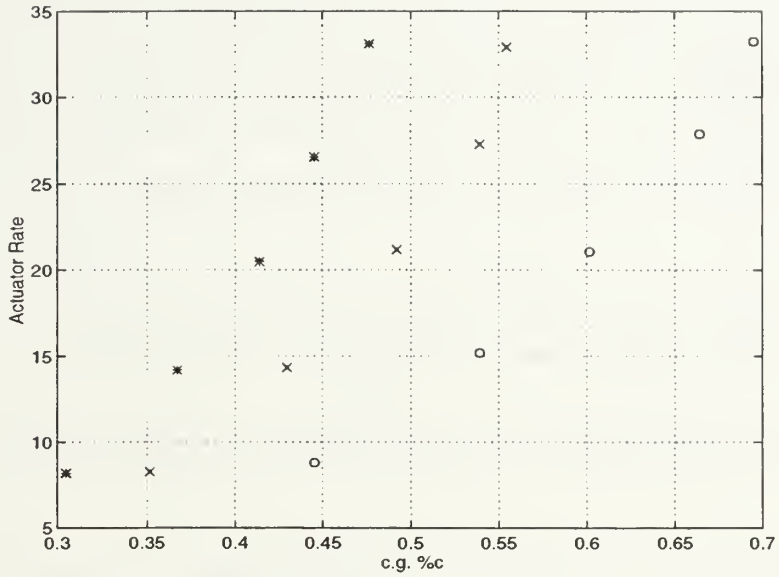


Figure 31. Increasing the wind-shear peak velocity moves the center of gravity limit forward.

On the other hand, figure 32 shows the sensitivity of the aft center of gravity location to changes in the penetration distance. The peak gust velocity was 45 feet per second. A penetration distance of 1, $1\frac{1}{2}$, and 2 cord lengths is shown. Choosing an actuator rate limit of 20 degrees per second again, every 50 percent increase in wind-shear penetration distance pushes the aft center-of-gravity station aft 5 percent.

On a percentage basis, the aft center of gravity limit is about twice as sensitive to changes in the peak wind-shear velocity as it is to changes in the penetration distance.

E. CONCLUSIONS

The sizing of the horizontal control surface(s) for HSCT is a difficult problem. Traditionally, aircraft definition has taken place apart from feedback considerations. Controller design was a derivative process, with little contribution to the aircraft

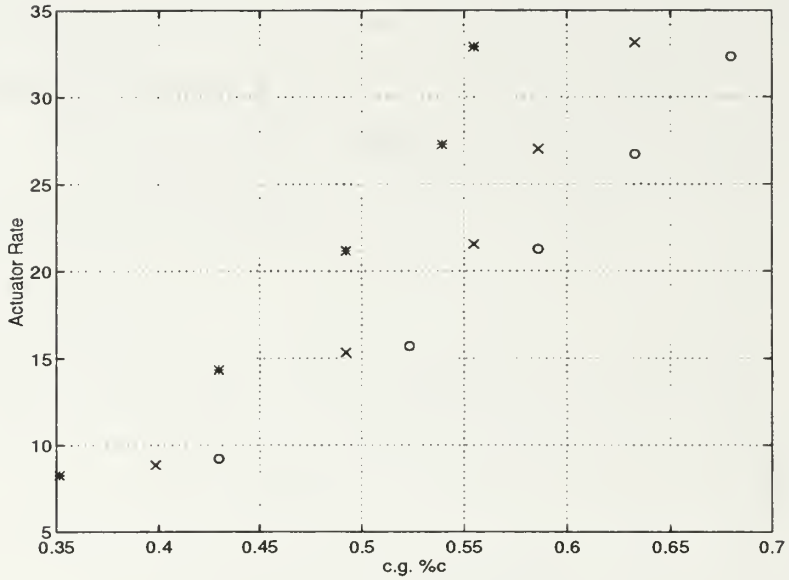


Figure 32. Increasing the wind-shear penetration distance moves the center-of-gravity limit aft.

definition. Of course, the aircraft were statically stable, and feedback control was viewed as a significant enhancement but not flight critical. This is an interesting example of an application where traditional methods simply lack the necessary tools to adequately define the aircraft configuration. Clearly, the aircraft definition will come first. However, it must be done with tools that provide a quantifiable knowledge of the impact on the demands of the feedback control system.

The most intuitive metric to select, capturing the demands of the feedback control system, is the peak actuator rate required. The inclusion of this metric adds an extra dimension to the tail sizing problem. The two dimensional tail sizing scissors plot is inadequate, and a natural extension is the *Tail Sizing Design Space.* A conventional scissors plot can be recovered by viewing the intersection of the *Tail Sizing Design Space* with a level plane. However, the three dimensional space allows the designer to see how sensitive the aft boundary is to changes in actuator rate. Clearly, there is value in knowing when small changes in the center-of-gravity location result in large changes in the actuator rate required.

While not solvable analytically, new efficient algorithms make the problem

tractable numerically. A new design tool is provided that provides the capability to quickly determine the *Tail Sizing Design Space* for a given aircraft configuration. This chapter demonstrates that many tail sizing problems can be formulated as LMIs, and exploit new interior point algorithms to obtain exact numerical solutions. The iterative nature of the design process requires a tool that can generate solutions in a timely manner. Using the *Tail Sizing Design Tool*, the user can make adjustments to the aircraft definition, and quantifiably asses the impact on the demands of the feedback control system.

In the first half of the chapter, two fundamental changes in aircraft definition were explored. Their influence on the *Tail Sizing Design Space* for a representative model of an HSCT was quantfied. The first was the use of canards in addition to a horizontal tail. The second was the effect of simple, flexible motion. This resulted in the testing the matrix of basic aircraft configurations shown in table E. Additionally, the effect of using a static, state-feedback controllers, or dynamic, full-state, output-feedback controllers, was investigated for each element in the matrix.

Rigid Body-Tail Only	Rigid Body-Tail with Canard
Flexible Body-Tail Only	Flexible Body-Tail with Canard

Table II. Matrix of Aircraft Definitions Tested

Numerical results suggest that canards provide a small benefit for a rigid body HSCT. Their use is more effective when used on a flexible body HSCT. The metric used to asses their effectiveness was the change in total horizontal control volume. In this example, an aeroelastic model realized a reduction in total horizontal control volume of approximately 30 percent through the use of a canard. The rigid-body model realized a reduction in total horizontal control volume of approximately 10 percent. The use of a dynamic, full-state, output-feedback controller did not improve the results.

In the second part of the chapter, a different dynamic constraint from the high angle-of-attack excursion was addressed. The dynamic constraint was the penetration of a severe vertical wind-shear. The definition of the wind-shear profile was open. Therefore, the effect of changes in the wind-shear profile on the *Tail Sizing Design Space* for a rigid-body HSCT was explored. In this example, a boundary of the *Tail Sizing Design Space* was found to be twice as sensitive to changes in the peak level of the gust then to changes in the distance to penetrate the gust.

III. INTEGRATED GUIDANCE AND CONTROL

A. INTRODUCTION

In a great number of envisioned mission scenarios, Autonomous Vehicles will be required to follow inertial reference trajectories accurately in 3-D space. See, for example, [Ref. 36, 9, 11] and the references therein. Similar requirements emerge from the recent work at NASA on the descent trajectory synthesis for air traffic control [Ref. 43]. To achieve this goal, the following systems must be designed and implemented on board autonomous vehicles (AVs): i) *navigation*, to provide estimates of linear and angular positions and velocities of the vehicle, ii) *guidance*, to process navigation/inertial reference trajectory data and output set-points for the vehicle's (body) velocity and attitude, and iii) *control*, to generate the actuator signals that are required to drive the actual velocity and attitude of the vehicle to the values commanded by the guidance scheme.

The advent of GPS (Global Positioning System) has afforded AV systems engineers a powerful new means of obtaining accurate navigation data that is required for precise tracking of given inertial trajectories. However, traditional guidance and control schemes used to steer the vehicle along such trajectories may prove inadequate in the case where frequent heading changes are required, or in the presence of shifting wind [Ref. 35]. Traditionally, such systems are designed separately, using well established design methods for control, and simple strategies such as line of sight (LOS) for guidance. See [Ref. 20] and [Ref. 29] for interesting applications to underwater and air vehicles, respectively. During the design phase, the control system is usually designed with sufficiently large bandwidth to track the commands that are expected from the guidance system. However, since the two systems are effectively coupled, stability and adequate performance of the combined system about nominal trajectories are not guaranteed [Ref. 35]. In practice, this problem can be resolved by

judicious choice of guidance law parameters (such as the "visibility distance" in LOS strategy), based on extensive computer simulations. Even when stability is obtained, however, the resulting strategy leads to finite trajectory tracking errors, the magnitude of which depends on the type of trajectory to be tracked (radius of curvature, vehicle's desired speed, etc.) [Ref. 35].

This chapter proposes a new methodology for the design of guidance and control systems for AVs, whereby the two systems are designed simultaneously. This methodology has two main advantages over traditional ones: i) the resulting trajectory tracking system achieves zero steady state tracking error about any trimming trajectory, and ii) the design methodology explicitly addresses the problem of stability of the combined guidance and control systems.

Earlier work based on this approach can be found in [Ref. 15, 42, 25]. In particular, the authors introduce a methodology for the design of controllers for UAVs to track inertial trajectories that are given in space and time coordinates. The trajectories considered are equilibrium (also known as trimming) trajectories of AVs, which are helices parameterized by the vehicle's linear speed, yaw rate and flight path angle. Furthermore, in [Ref. 15, 42, 25] it is shown the linearization of the vehicle error dynamics and kinematics about any trimming trajectory is time-invariant. Thus, the problem of designing integrated guidance and control systems for AVs to accurately track trimming trajectories can be solved by using tools that borrow from gain scheduling control theory, particularly those reported in [Ref. 27]. Within the framework of [Ref. 27], the vehicle's linear speed, yaw rate, and flight path angle play the role of scheduling variables that interpolate the parameters of linear controllers designed for a finite number of representative trimming trajectories. The results introduced in [Ref. 27] on the D-implementation of gain scheduled controllers can then be used to obtain a combined guidance/control system such that the properties of the linear designs are recovered locally, about each trimming trajectory. An interesting and very important consequence of the D-implementation is that it leads naturally to

a controller structure where the only exogenous commands required are the desired linear inertial position and the yaw rate, thus avoiding the need to feedforward the trimming conditions for the remaining state variables and control inputs.

However, the technique presented in [Ref. 15, 42, 25] has a shortcoming which may be of concern for UAVs and AUVs when tracking trimming trajectories in the presence of changing air and water mass. Since the controllers described in those references achieve accurate tracking of trajectories defined in terms of space and time coordinates, the relative speed of the vehicle with respect to the air cannot be controlled externally, its value being computed internally as a function of the tracking error. In practice, this may lead to unacceptable performance in the presence of winds, since the change in the airspeed of the vehicle may result in stall or structural damage.

Clearly, eliminating the time coordinate in the trajectory definition and using the vehicle attitude to null out trajectory errors while maintaining constant airspeed should resolve this problem. A similar approach has been introduced in a number of publications on robot control. Of particular interest is the work reported in [Ref. 40], where the subject of path following control for wheeled robots is addressed. See also [Ref. 35] for a detailed analysis of the stability of an autonomous underwater vehicle about nominal trajectories in the horizontal plane.

In this chapter, these ideas are formalized within the basic framework for 3-D trajectory tracking controller system design developed in [Ref. 15, 42, 25]. Using the concepts outlined in [Ref. 40], the linear position of an AV is given in terms of its location with respect to the closest point on a desired trajectory, together with the arc length of an imaginary curve traced along that trajectory. Tracking of a trimming trajectory by the vehicle at a fixed speed is then converted into the problem of driving a generalized error vector - which implicitly includes the distance to the trajectory - to zero. Moreover, it is shown that the linearization of the generalized error dynamics about the corresponding trimming path is time-invariant. Using these results, the problem of trajectory tracking is posed and solved in the framework

of gain scheduled control theory, leading to a new technique for integrated design of guidance and control systems for AVs. The chapter summarizes the resulting design methodology, and illustrates its application to the design and implementation of a nonlinear trajectory tracking controller for the UAV *Frog* [Ref. 25]. Numerical simulations using a full set of nonlinear equations of motion of the vehicle show the effectiveness of the proposed techniques.

The subject of trajectory tracking has also been addressed in the literature on control of nonholonomic vehicles. In [Ref. 44], the problem of tracking a nominal trajectory by a nonlinear system is considered. The key idea includes linearizing the nonlinear system along the trajectory, then using the resulting time varying linearization to obtain a time varying, state-feedback controller that locally exponentially stabilizes the system along the trajectory. The paper includes examples of applications of the proposed technique to a mobile robot and a front wheel drive car. The nominal trajectories considered in [Ref. 44] are not restricted to be trimming trajectories. However, all the examples presented consider the case of trimming trajectories only. Another approach is used in [Ref. 19], where a tracking problem for a surface marine vessel is considered. Here the authors use feedback linearization with dynamic extension to obtain a controller to track trajectories that consist of lines and arcs of circles (a special case of trimming trajectories in the plane).

The solution to the trajectory tracking problem proposed in this chapter differs considerably from the ones introduced in [Ref. 44, 19]. Here, the key idea is to reduce the problem to the design of a tracking controller for a linear-time invariant plant utilizing a simple nonlinear transformation that inverts the vehicle kinematics. This poses no robustness concerns since kinematics are usually well known, particularly in the case of air and underwater vehicles. It is important to point out that the application of the nonlinear transformation results in a nonlinear plant, whose linearization along trimming trajectories is time-invariant. Once in the linear setting, the designer is free to choose his favorite control synthesis technique to achieve the

desired closed-loop performance and robustness. The chapter provides a simple algorithm for implementing the linear controller on the nonlinear plant such that the properties of linear controller are preserved along each trajectory. This is in contrast to the approach in [Ref. 44], where the problem is reduced to that of designing an exponentially stabilizing state-feedback controller for a linear time-varying system. This leads to a controller design that is problem specific and does not address the issues of performance and robustness. On the other hand, in [Ref. 19] the authors point out that extending to air vehicles the feedback linearization technique use for trajectory tracking of the surface craft is difficult due to the unstable zero dynamics that are characteristic of aircraft.

The chapter is organized as follows. Section B develops the rigid body dynamics and introduces appropriate notation. Section C introduces the error dynamics necessary to solve the problem. Section D formulates and solves the problem of integrated guidance and control of UAVs for the class of trajectories introduced in Section C. Section E describes an application to the integrated guidance and control of the UAV *Frog*. Finally, Section F contains the main conclusions.

B. RIGID BODY DYNAMICS

This section begins with a review of the equations governing the motion of a rigid body, and specifically of an unmanned air vehicle. Notation familiar in robotics [Ref. 10], but not commonly used in the field of aircraft dynamics, is introduced. It is hoped that the familiar environment of the equations of motion of a rigid body will make the reader comfortable with the notation. It will be used extensively later in developing an integrated guidance and control algorithm.

Let $\{I\}$ denote an inertial reference frame. For the purposes of this development, the rotation of the earth and its associated Coriolis' force can be ignored. Thus, a local tangent plane reference frame will be considered an inertial reference frame. Let $\{B\}$ denote a body coordinate frame that is fixed with respect to the body

of the vehicle. Finally, let $\{W\}$ denote a stability coordinate frame aligned with the velocity vector of the vehicle.

Free vectors can be expressed in whatever reference frame is most convenient. The velocity of the vehicle with respect to $\{I\}$ is a free vector. When resolved in $\{B\}$, it will be described by

$$V = \begin{bmatrix} u & v & w \end{bmatrix}^T. \tag{III.1}$$

Free vectors can be transformed from one frame to another via the rotation matrix describing the relative orientation of the two frames. For example, it is common to describe the orientation of $\{B\}$ with respect to $\{I\}$, by the Euler angles $(\phi,\ \theta,\ \psi)$. Let

$$\Lambda = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix},$$

then ${}^B_I R(\Lambda)$, or simply ${}^B_I R$, is the rotation matrix from $\{I\}$ to $\{B\}$, and given by

$${}^B_I R = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \sin(\psi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \sin(\phi)\cos(theta) \\ \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(psi) - \sin(\phi)\sin(psi) & \cos(\phi)\cos(\theta) \end{bmatrix}. \tag{III.2}$$

As an example, consider the position P of the vehicle. Its velocity, $\frac{d}{dt}P$, in $\{I\}$ can be expressed as

$$\frac{d}{dt}P = {}^I_B RV.$$

Another common rotation matrix relates the orientation of $\{B\}$ to the stability axis of the vehicle $\{W\}$. The vehicle's angle of attack, α, and side-slip angle, β, define the orientation of $\{W\}$ with respect to $\{B\}$. The associated rotation matrix is termed ${}^B_W R(\alpha, \beta)$. As an example, if V is the velocity of the vehicle with respect

to $\{I\}$, resolved in $\{B\}$, then

$$\begin{bmatrix} \|V\| \\ 0 \\ 0 \end{bmatrix} = {}^{W}_{B}R(\alpha,\beta)V$$

is the same vector resolved in $\{W\}$.

The angular velocity of $\{B\}$, with respect to $\{I\}$, resolved in $\{B\}$ will be represented by Ω. It is the set of familiar body-fixed rotation rates, roll rate (p), pitch rate (q), and yaw rate (r). The body-fixed rotation rates are related to the Euler angles by

$$\begin{aligned} \begin{bmatrix} p \\ q \\ r \end{bmatrix} &= \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \frac{d}{dt}\phi \\ \frac{d}{dt}\theta \\ \frac{d}{dt}\psi \end{bmatrix}, \\ &= \mathcal{Q}^{-1}(\Lambda)\dot{\Lambda}. \end{aligned} \tag{III.3}$$

Inverting (III.3), the time rate of change of Λ is related to Ω by the matrix $\mathcal{Q}(\Lambda)$, or simply $\mathcal{Q}$, as

$$\begin{aligned} \frac{d}{dt}\Lambda &= \mathcal{Q}\Omega, \\ \mathcal{Q}(0) &= I. \end{aligned} \tag{III.4}$$

Using this notation, an application of Newton's Law to the linear and angular motion of the vehicle yields

$$\begin{aligned} \frac{d}{dt}V &= -\Omega \times V + \frac{F}{m}, \\ \frac{d}{dt}\Omega &= \mathcal{I}_B^{-1}\Omega \times \mathcal{I}_B\Omega + \mathcal{I}_B^{-1}N, \end{aligned} \tag{III.5}$$

where F and N are the total external force and moment acting on the vehicle resolved in $\{B\}$, m is the body mass, and $\mathcal{I}_B$ is the inertia tensor of the body resolved in $\{B\}$.

Standard practice is to separate out the contribution of gravity, (G), from the force and moment terms. The remaining terms are expanded in a first order Taylor

series expansion about some choice of state variables and control inputs. A convenient choice of state variables is $(u,\ \beta,\ \alpha,\ p,\ q,\ r)$, and standard control inputs are elevator (δe), aileron (δa), rudder (δr) and throttle (δt). Applying this to (III.5) results in

$$\begin{aligned}\frac{d}{dt}V &= -\Omega \times V + {}^{B}_{I}RG + \frac{1}{2m}\rho\|V\|^2 \mathrm{s}\mathcal{R}_w(\alpha,\beta)\left\{\begin{bmatrix} C_{D_0} \\ C_{Y_0} \\ C_{L_0}\end{bmatrix} + \begin{bmatrix} C_{D_u} & 0 & C_{D_\alpha} \\ 0 & C_{Y_\beta} & 0 \\ C_{L_u} & 0 & C_{L_\alpha}\end{bmatrix}\begin{bmatrix} u \\ \beta \\ \alpha\end{bmatrix}\right. \\ &\left. + \begin{bmatrix} 0 & C_{D_q} & 0 \\ C_{Y_p} & 0 & C_{Y_r} \\ 0 & C_{L_q} & 0\end{bmatrix}\Omega + \begin{bmatrix} C_{D_{\delta t}} & C_{D_{\delta e}} & 0 & 0 \\ 0 & 0 & C_{Y_{\delta a}} & C_{Y_{\delta r}} \\ C_{L_{\delta t}} & C_{L_{\delta e}} & 0 & 0\end{bmatrix}\begin{bmatrix} \delta t \\ \delta e \\ \delta a \\ \delta r\end{bmatrix}\right\}, \qquad \text{(III.6)}\end{aligned}$$

$$\begin{aligned}\frac{d}{dt}\Omega &= -\mathcal{I}_B^{-1}\Omega \times \mathcal{I}_B\Omega + \mathcal{I}_B^{-1}\left\{\frac{1}{2}\rho\|V\|^2\begin{bmatrix} sb & 0 & 0 \\ 0 & s\bar{c} & 0 \\ 0 & 0 & sb\end{bmatrix}\mathcal{R}_w(\alpha,\beta)\left\{\begin{bmatrix} C_{l_0} \\ C_{M_0} \\ C_{N_0}\end{bmatrix}\right.\right. \\ &+ \begin{bmatrix} 0 & C_{l_\beta} & 0 \\ C_{M_u} & 0 & C_{M_\alpha} \\ 0 & C_{l_\beta} & 0\end{bmatrix}\begin{bmatrix} u \\ \beta \\ \alpha\end{bmatrix} + \begin{bmatrix} C_{l_p} & 0 & C_{l_r} \\ 0 & C_{M_q}0 & \\ C_{N_p} & 0 & C_{N_r}\end{bmatrix}\begin{bmatrix} \frac{b}{2\|V\|} & 0 & 0 \\ 0 & \frac{\bar{c}}{2\|V\|} & 0 \\ 0 & 0 & \frac{b}{2\|V\|}\end{bmatrix}\Omega \\ &\left.\left. + \begin{bmatrix} 0 & 0 & C_{l_{\delta a}} & C_{l_{\delta r}} \\ C_{M_{\delta t}} & C_{M_{\delta e}} & 0 & 0 \\ 0 & 0 & C_{N_{\delta a}} & C_{N_{\delta r}}\end{bmatrix}\begin{bmatrix} \delta t \\ \delta e \\ \delta a \\ \delta r\end{bmatrix}\right\}\right\}, \qquad \text{(III.7)}\end{aligned}$$

where

$$
\begin{aligned}
s &:= \text{wing reference area,} \\
\bar{c} &:= \text{wing mean chord,} \\
m &:= \text{mass,} \\
\mathcal{I}_B &:= \text{aircraft's inertia tensor resolved in } \{B\}, \\
\rho &:= \text{air density,} \\
b &:= \text{wing span,} \\
D,\ Y,\ L &:= \text{normalized drag, lateral force and lift,} \\
l,\ M,\ N &:= \text{normalized roll pitch and yaw moments,} \\
C_{x_0} &:= \text{normalized nominal force or moment coefficient,} \\
C_{x_y} &:= \text{stability derivative:} \frac{\partial x}{\partial y}.
\end{aligned}
$$

Notice that some symbols in (III.6) and (III.7) have multiple meanings, and attention must be paid to the context in which they are used. The terminology is standard in aircraft dynamics.

A convenient grouping of terms in (III.6) and (III.7) is to let $\mathcal{H}$ be a vector valued function of the control inputs, $\mathcal{I}$ be a vector valued function of all the terms affine in $\mathcal{H}$, and $\mathcal{F}$ be a vector valued function of all the remaining terms. Then, equations III.6 and III.7 can be compactly expressed in terms of these functions. Subscripts are used to indicate the correspondence of the function to the appropriate linear or angular motion equation. As an example,

$$
\mathcal{I}_V := \frac{1}{2m}\rho\|V\|^2 \mathbf{s}\mathcal{R}_w(\alpha,\beta)\begin{bmatrix} C_{D_{\delta t}} & C_{D_{\delta e}} & 0 & 0 \\ 0 & 0 & C_{Y_{\delta a}} & C_{Y_{\delta r}} \\ C_{L_{\delta t}} & C_{L_{\delta e}} & 0 & 0 \end{bmatrix}. \tag{III.8}
$$

Then, the complete equations of motion of the vehicle are expressed in this notation

as

$$\mathcal{G} = \begin{cases} \frac{d}{dt} V &= \mathcal{F}_V(V,\Omega,\Lambda) + \mathcal{I}_V(V,\Omega)\mathcal{H}(V,\Omega,U), \\ \frac{d}{dt} \Omega &= \mathcal{F}_\Omega(V,\Omega,\Lambda) + \mathcal{I}_\Omega(V,\Omega)\mathcal{H}(V,\Omega,U), \\ \frac{d}{dt} P &=^I_B RV, \\ \frac{d}{dt} \Lambda &= \mathcal{Q}\,\Omega. \end{cases} \tag{III.9}$$

C. GENERALIZED ERROR DYNAMICS

1. Trimming Trajectories

The objective of the guidance and control systems is to steer an autonomous vehicle along prescribed inertial trajectories $P_C \in \mathcal{R}^3$. Furthermore, we require that the vehicle be trimmed along any such trajectory. Let $\{C\}$ define the coordinate system attached to the vehicle, and let Λ_C define the desired inertial orientation of $\{C\}$. The coordinate system $\{C\}$ represents the desired inertial orientation of the vehicle along P_C. Therefore, at trim $\{B\} = \{C\}$. Next, we define the set $\mathcal{E}$ of the *trimming* trajectories about which the vehicle is expected to operate:

$$\mathcal{E} := \left\{ \begin{bmatrix} P_C \\ \Lambda_C \end{bmatrix} : \begin{array}{l} \frac{d}{dt} P_C =^I_C RV_C, \\ \frac{d}{dt} \Lambda_C = \mathcal{Q}(\Lambda_C)\,\Omega_C =: \mathcal{Q}_C\Omega_C, \\ \mathcal{F}_V(V_C,\Omega_C,\Lambda_C) + \mathcal{I}_V(V_C,\Omega_C)\mathcal{H}(V_C,\Omega_C,U_c) = 0, \\ \mathcal{F}_\Omega(V_C,\Omega_C,\Lambda_C) + \mathcal{I}_\Omega(V_C,\Omega_C)\mathcal{H}(V_C,\Omega_C,U_c) = 0, \end{array} \right\} \tag{III.10}$$

where V_C, Ω_C and U_c denote the trimming values of V, Ω and U, respectively, and Λ_C denotes the vector of Euler angles that describe the orientation of $\{C\}$ with respect to $\{I\}$.

From the definition of $\mathcal{E}$ and equations III.9, it can be concluded [Ref. 13] that trimming trajectories $P_C \in \mathcal{E}$ correspond to *helices*:

$$\frac{d}{dt}\Lambda_C = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi}_c \end{bmatrix}, \tag{III.11}$$

$$\frac{d}{dt}P_C = \begin{bmatrix} -v_c\cos(\gamma_c)\sin(\dot{\psi}_c t) \\ v_c\cos(\gamma_c)\cos(\dot{\psi}_c t) \\ -v_c\sin(\gamma_c) \end{bmatrix}, \tag{III.12}$$

where

$$\begin{aligned} \dot{\psi}_c &= \text{desired turn rate,} \\ v_c &= \text{desired inertial velocity,} \\ \gamma_c &= \text{desired flight path angle.} \end{aligned}$$

Thus, the trimming trajectories can be parameterized by the vector

$$\eta_c = [v_c, \dot{\psi}_c, \gamma_c]^T \in \mathcal{R}^3. \tag{III.13}$$

In fact, given η_c, we can determine the trimming values for V_C, Ω_C and U_C which will be important later for the controller synthesis.

Integrating (III.12) we obtain

$$P_C(t) = \begin{bmatrix} \frac{v_c\cos(\gamma_c)}{\dot{\psi}_c}\cos(\dot{\psi}_c t) \\ \frac{v_c\cos(\gamma_c)}{\dot{\psi}_c}\sin(\dot{\psi}_c t) \\ -v_c\sin(\gamma_c)t \end{bmatrix}. \tag{III.14}$$

Equation III.14 indicates that the radius of the helix is

$$b_c := \frac{v_c\cos(\gamma_c)}{\dot{\psi}_c}, \tag{III.15}$$

and the climb rate is

$$\dot{h}_c := -v_c\sin(\gamma_c). \tag{III.16}$$

Using elementary ideas from differential geometry [Ref. 3], equation III.14 can be reparameterized by considering the arclength s defined as

$$\begin{aligned} s &:= \int \|\frac{d}{dt}P_C(t)\|dt, \\ &= \int v_c dt. \end{aligned} \tag{III.17}$$

Although the symbol for arclength is the same as that used in other sections for the Laplace transform variable, it is hoped that the context will make the distinction clear. Then, using (III.15),(III.16) and (III.17), equation III.14 is written as

$$P_C(s) = \begin{bmatrix} b_c \cos(\dot{\psi}_c \frac{s}{v_c}) \\ b_c \sin(\dot{\psi}_c \frac{s}{v_c}) \\ -\dot{h}_c \frac{s}{v_c} \end{bmatrix}. \tag{III.18}$$

Two parameters useful in describing the helix, $P_C(s)$, are its curvature and its torsion. The curvature is defined in terms of the parameters in η_c as

$$\kappa(s) = \frac{\dot{\psi}_c cos(\gamma_c)}{v_c}, \tag{III.19}$$

and the torsion as

$$\begin{aligned} \tau(s) &= \frac{\dot{h}_c \dot{\psi}_c}{v_c^2}, \\ &= \frac{\dot{\psi}_c \sin\gamma_c}{v_c}. \end{aligned} \tag{III.20}$$

2. Trimming Trajectory Coordinate System

Now, let $\{A\}$ denote a Frenet frame attached to $P_C(s)$ [Ref. 3]. The x axis of the Frenet frame is termed T. It is a unit vector tangent to $P_C(s)$ at s, and it points in the direction of motion on $P_C(s)$. Its components are defined as

$$\begin{aligned} T(s) &= \frac{dP(s)}{ds}, \\ &= \begin{bmatrix} -\frac{b_c \dot{\psi}_c}{v_c} \sin(\dot{\psi}_c \frac{s}{v_c}) \\ \frac{b_c \dot{\psi}_c}{v_c} \cos(\dot{\psi}_c \frac{s}{v_c}) \\ -\frac{\dot{h}_c}{v_c} \end{bmatrix}, \\ &= \begin{bmatrix} -\cos(\gamma_c) \sin(\dot{\psi}_c \frac{s}{v_c}) \\ \cos(\gamma_c) \cos(\dot{\psi}_c \frac{s}{v_c}) \\ -\sin(\gamma_c) \end{bmatrix}. \end{aligned} \tag{III.21}$$

The y axis is termed N and is perpendicular to T. It is a unit vector pointing toward the center of the helix, and it is defined as

$$\begin{aligned} N(s) &= \frac{dT(s)}{ds}/\kappa(s), \\ &= \begin{bmatrix} -\cos(\dot{\psi}_c \frac{s}{v_c}) \\ -\sin(\dot{\psi}_c \frac{s}{v_c}) \\ 0 \end{bmatrix}. \end{aligned} \tag{III.22}$$

The z axis is termed B, and is orthonormal to T and N. It is defined according to the right hand rule, $B(s) = T(s) \times N(s)$:

$$B(s) = \begin{bmatrix} \sin(\gamma_c)\sin(\dot{\psi}_c \frac{s}{v_c}) \\ -\sin(\gamma_c)\cos(\dot{\psi}_c \frac{s}{v_c}) \\ \cos(\gamma_c) \end{bmatrix}. \tag{III.23}$$

The rotation matrix from $\{A\}$ to $\{I\}$ can be defined by the vectors, T, N, and B as

$${}^I_A\mathcal{R} = [T \;\; N \;\; B]. \tag{III.24}$$

Let ${}^A\Omega$ be the angular velocity of $\{A\}$ with respect to $\{I\}$, resolved in $\{A\}$. It can be shown that ${}^A\Omega$ is equal to $[\tau\dot{s} \;\; 0 \;\; \kappa\dot{s}]^T$. Using ${}^A\Omega$ and the Serret-Frenet Theorem [Ref. 3], the time rate of change of ${}^I_A\mathcal{R}$ is

$$\begin{aligned} \frac{d({}^I_A\mathcal{R})}{dt} &= \frac{d({}^I_A\mathcal{R})}{ds}\dot{s}, \\ &= [T \;\; N \;\; B] \begin{bmatrix} 0 & -\kappa & 0 \\ \kappa & 0 & -\tau \\ 0 & \tau & 0 \end{bmatrix} \dot{s}, \\ &= {}^I_A\mathcal{R}\mathcal{S}({}^A\Omega), \end{aligned} \tag{III.25}$$

where $\mathcal{S}(Y)$ is a skew symmetric matrix defined by [Ref. 10],

$$\begin{aligned} \mathcal{S}(Y) &= Y\times, \\ &= \begin{bmatrix} 0 & -\kappa\dot{s} & 0 \\ \kappa\dot{s} & 0 & -\tau\dot{s} \\ 0 & \tau\dot{s} & 0 \end{bmatrix}. \end{aligned}$$

Let $P(s_0)$ denote a point on the trajectory $P_C \in \mathcal{E}$. Define the error vector

$$M_E \;:=\; {}^A_I\mathcal{R}(P - P(s_0)). \tag{III.26}$$

We will call $P_C(s_0)$ a projection of P onto $P_C \in \mathcal{E}$ when M_E has the following form:

$$M_E = [0 \;\; y \;\; z]^T.$$

From the definition of M_E, $P_C(s_0)$ is also the point on P_C nearest to P. (For the discussion of when such a point can be uniquely determined, see [Ref. 40]). Let $P_C(s_0)$ be the projection of P onto $P_C \in \mathcal{E}$. Then, using the definitions of M_E and $\{A\}$, we obtain

$$\begin{aligned}
\frac{d}{dt}P &= {}^I_B RV \\
&= \frac{d}{dt}P_C + \frac{d}{dt}({}^I_A\mathcal{R}M_E), \\
&= {}^I_A\mathcal{R}\begin{bmatrix} \dot{s} \\ 0 \\ 0 \end{bmatrix} + {}^I_A\,\mathcal{R}\mathcal{S}({}^A\Omega)M_E + {}^I_A\,\mathcal{R}\begin{bmatrix} 0 \\ \dot{y} \\ \dot{z} \end{bmatrix}.
\end{aligned} \tag{III.27}$$

Therefore,

$$\begin{aligned}
{}^A_B RV &= {}^A_I R^I_B RV, \\
&= {}^A_I R^I_A \mathcal{R}\begin{bmatrix} \dot{s} \\ 0 \\ 0 \end{bmatrix} + {}^A_I\,R^I_A\mathcal{R}\mathcal{S}({}^A\Omega)M_E + {}^A_I\,R^I_A\mathcal{R}\begin{bmatrix} 0 \\ \dot{y} \\ \dot{z} \end{bmatrix}, \\
&= \begin{bmatrix} \dot{s} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \mathcal{S}({}^A\Omega)M_E, \\
&= \begin{bmatrix} \dot{s} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} -y\kappa\dot{s} \\ -z\tau\dot{s} \\ y\tau\dot{s} \end{bmatrix},
\end{aligned}$$

$$= \begin{bmatrix} 1-y\kappa & 0 & 0 \\ z\tau & 1 & 0 \\ y\tau & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{s} \\ \dot{y} \\ \dot{z} \end{bmatrix}, \tag{III.28}$$

or

$$\begin{bmatrix} \dot{s} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 1-y\kappa & 0 & 0 \\ z\tau & 1 & 0 \\ y\tau & 0 & 1 \end{bmatrix}^{-1} {}^{A}_{B}\mathcal{R}V. \tag{III.29}$$

3. Generalized Error Vector

The design of an integrated guidance and control system for the plant $\mathcal{G}$ in (III.9) and the set $\mathcal{E}$ involves obtaining linear models for $\mathcal{G}$ along the trajectories in $\mathcal{E}$. These models will necessarily be time-varying in the state space coordinates used in (III.9). It turns out, however, that an appropriate coordinate system exists where the linearization of the plant $\mathcal{G}$ along any trajectory $P_C \in \mathcal{E}$ is time-invariant. This coordinate system is discussed next. Let $P_C, \Lambda_C \in \mathcal{E}$ be given. Define

$$\begin{cases} V_E &= V - V_C, \\ \Omega_E &= \Omega - \Omega_C, \\ P_E &= \begin{bmatrix} s & y & z \end{bmatrix}^T, \\ \Lambda_E &= \mathcal{Q}^{-1}[\Lambda - \Lambda_C], \end{cases} \tag{III.30}$$

which can be interpreted as the generalized error vector between the vehicle state and the trajectory in $\mathcal{E}$. Simple physical considerations show that the problem of following a trimming trajectory $P_C \in \mathcal{E}$ at a fixed speed V_C is equivalent to driving the generalized error vector to zero.

In order to have a complete description of the error dynamics, we need to derive an expression for the time rate of change of the generalized error vector, (III.30). Expressions (III.9) and (III.29) suffice to describe the derivatives of the first three terms in (III.30), since V_C and Ω_C are constant along trimming trajectories. The

fourth term, Λ_E, is the only term that requires an explanation. Recall,

$$\Lambda_E \;=\; \mathcal{Q}^{-1}\left[\Lambda - \Lambda_C\right]. \tag{III.31}$$

Therefore,

$$\frac{d}{dt}\Lambda_E \;=\; \mathcal{Q}^{-1}\left(\frac{d}{dt}\Lambda - \frac{d}{dt}\Lambda_C\right) + \frac{d}{dt}\mathcal{Q}^{-1}\left[\Lambda - \Lambda_C\right]. \tag{III.32}$$

Since

$$\frac{d}{dt}\Lambda \;=\; \mathcal{Q}\Omega,$$

and

$$\frac{d}{dt}\Lambda_C \;=\; \mathcal{Q}_C\Omega_C,$$

we obtain

$$\frac{d}{dt}\Lambda_E \;=\; \Omega - \mathcal{Q}^{-1}\mathcal{Q}_C\Omega_C + \frac{d}{dt}\mathcal{Q}^{-1}\left[\Lambda - \Lambda_C\right]. \tag{III.33}$$

Therefore, the generalized error dynamics are described by the system of equations

$$\mathcal{G}_E = \left\{ \begin{array}{rcl} \frac{d}{dt}V_E &=& \mathcal{F}_{V_E}(V_E,\Omega_E,\Lambda_E) \;+\; \mathcal{I}_{V_E}(V_E,\Omega_E)\mathcal{H}(V_E,\Omega_E,U), \\ \frac{d}{dt}\,\Omega_E &=& \mathcal{F}_{\Omega_E}(V_E,\Omega_E,\Lambda_E) \;+\; \mathcal{I}_{\Omega_E}(V_E,\Omega_E)\mathcal{H}(V_E,\Omega_E,U), \\ \frac{d}{dt}P_E &=& \left[\begin{array}{ccc} 1-y\kappa & 0 & 0 \\ z\tau & 1 & 0 \\ y\tau & 0 & 1 \end{array}\right]^{-1} {}^{A}_{B}\mathcal{R}V, \\ \frac{d}{dt}\Lambda_E &=& \Omega_E - \Omega_C - \mathcal{Q}^{-1}\mathcal{Q}_C\Omega_C + \dot{\mathcal{Q}}^{-1}\mathcal{Q}\Lambda_E, \end{array} \right. \tag{III.34}$$

where

$$\mathcal{F}_{V_E}(V_E,\Omega_E,\Lambda_E) = \mathcal{F}_V(V_E+V_C,\Omega_E+\Omega_C,\mathcal{Q}\Lambda_E+\Lambda_C),$$

and similarly for $\mathcal{I}_{V_E}$, $\mathcal{F}_{\Omega_E}$ and $\mathcal{I}_{\Omega_E}$.

An important consequence of this choice of error dynamics is that the linearization of equations III.34 along any trajectory $P_C \in \mathcal{E}$ is *time-invariant.* In order to derive the linearization of (III.34), we require the following identity.

Identity 1: *Let* ${}^I_B R$ *be the rotation matrix from* $\{B\}$ *to* $\{I\}$, *and let* $\mathcal{Q}$ *satisfy* $\frac{d}{dt}(\Lambda) = \mathcal{Q}\Omega$. *Then for any vector* X *in* $\mathcal{R}^3$

$$\frac{d}{d\Lambda}({}^I_B RX) = -{}^I_B R\mathcal{S}(X)\mathcal{Q}^{-1}, \tag{III.35}$$

and

$$\frac{d}{d\Lambda}({}^B_I RX) = \mathcal{S}({}^B_I RX)\mathcal{Q}^{-1}. \tag{III.36}$$

Proof: First we observe that [Ref. 10]:

$$\frac{d}{dt}({}^I_B R) = {}^I_B R\mathcal{S}(\Omega). \tag{III.37}$$

Next, from the definition of a cross-product we get

$$\mathcal{S}(X)Y = -\mathcal{S}(Y)X \tag{III.38}$$

for any $X, Y \in \mathcal{R}^3$. Now, using equations III.37, III.38, and the fact that X is a constant vector, we obtain

$$\begin{aligned} \frac{d}{dt}({}^I_B RX) &= \frac{d}{dt}({}^I_B R)X + {}^I_B R\frac{d}{dt}X, \\ &= {}^I_B R\mathcal{S}(\Omega)X, \\ &= -{}^I_B R\mathcal{S}(X)\Omega. \end{aligned} \tag{III.39}$$

Next, observe that

$$\begin{aligned} \frac{d}{dt}({}^I_B RX) &= \frac{d}{d\Lambda}({}^I_B RX)\frac{d}{dt}\Lambda, \\ &= \frac{d}{d\Lambda}({}^I_B RX)\mathcal{Q}\,\Omega. \end{aligned} \tag{III.40}$$

Equation III.35 now follows by comparing equations III.39 and III.40.

To obtain equation III.36 note,

$$({}^B_I R)^{-1} = ({}^I_B R),$$

thus

$$\begin{aligned} \frac{d}{dt}({}^{B}_{I}R) &= -{}^{B}_{I}R\frac{d}{dt}({}^{I}_{B}R){}^{B}_{I}R, \\ &= -\mathcal{S}(\Omega){}^{B}_{I}R. \end{aligned} \tag{III.41}$$

Therefore,

$$\begin{aligned} \frac{d}{dt}({}^{B}_{I}RX) &= -\mathcal{S}(\Omega){}^{B}_{I}RX, \\ &= \mathcal{S}({}^{B}_{I}RX)\Omega. \end{aligned} \tag{III.42}$$

Moreover, using the chain rule, it follows that

$$\begin{aligned} \frac{d}{dt}({}^{B}_{I}RX) &= \frac{d}{d\Lambda}({}^{B}_{I}RX)\frac{d}{dt}\Lambda, \\ &= \frac{d}{d\Lambda}({}^{B}_{I}RX)\mathcal{Q}\Omega. \end{aligned} \tag{III.43}$$

Equation III.36 follows readily from equations III.42 and III.43.

Finally, let $\Lambda = 0$, then

$${}^{I}_{B}\mathcal{R} = {}^{B}_{I}\mathcal{R} = \mathcal{Q} = I.$$

Now, using equations III.35 and III.36, we get

$$\begin{aligned} \frac{d}{d\Lambda}({}^{I}_{B}RX)|_{\Lambda=0} &= \frac{d}{d\Lambda}({}^{B}_{I}RX)|_{\Lambda=0}, \\ &= S({}^{B}_{I}RX)\mathcal{Q}^{-1}, \\ &= \mathcal{S}(X). \end{aligned} \tag{III.44}$$

■

Since any trajectory $P_C \in \mathcal{E}$ defines a trim condition for equations III.6 and III.7, the linearization of (III.6) and (III.7) is naturally time invariant. Introducing the following notation,

$$\begin{aligned} \mathcal{A}^{X}_{Y} &\equiv \frac{\partial}{\partial Y}[\mathcal{F}_X(.) + \mathcal{I}_X(.)\mathcal{H}(.)], \\ \mathcal{B}_Z &\equiv \frac{\partial}{\partial Z}[\mathcal{I}_X(.)\mathcal{H}(.)], \end{aligned}$$

the linearization of (III.6) and (III.7) is written as

$$\begin{aligned}\frac{d}{dt}\delta V_E &= \mathcal{A}_V^V \delta V_E + \mathcal{A}_\Omega^V \delta\Omega_E + \mathcal{A}_\Lambda^V \delta\Lambda_E + \mathcal{B}_V \delta U,\\ \frac{d}{dt}\delta\Omega_E &= \mathcal{A}_V^\Omega \delta V_E + \mathcal{A}_\Omega^\Omega \delta\Omega_E + \mathcal{A}_\Lambda^\Omega \delta\Lambda_E + \mathcal{B}_\Omega \delta U.\end{aligned} \tag{III.45}$$

We need only to derive the expressions for $\frac{d}{dt}\delta P_E$, and $\frac{d}{dt}\delta\Lambda_E$ to show that they are time invariant.

Let

$$\begin{aligned}\mathcal{M} &= \begin{bmatrix} 1-y\kappa & 0 & 0 \\ z\tau & 1 & 0 \\ y\tau & 0 & 1 \end{bmatrix}^{-1}, \\ &= \begin{bmatrix} (1-y\kappa)^{-1} & 0 & 0 \\ -z\tau(1-y\kappa)^{-1} & 1 & 0 \\ -y\tau(1-y\kappa)^{-1} & 0 & 1 \end{bmatrix}.\end{aligned} \tag{III.46}$$

Then, from (III.34),

$$\begin{aligned}\frac{d}{dt}P_E &= \mathcal{M}(P_E)_B^A \mathcal{R}(\Lambda)V \\ &= \mathcal{M}_B^A \mathcal{R}V.\end{aligned} \tag{III.47}$$

Furthermore

$$\frac{d}{dt}\delta P_E = \frac{\partial}{\partial P_E}[\mathcal{M}_B^A \mathcal{R}V]|_c \delta P_E + \frac{\partial}{\partial \Lambda_E}[\mathcal{M}_B^A \mathcal{R}V]|_c \delta\Lambda_E + \frac{\partial}{\partial V}[\mathcal{M}_B^A \mathcal{R}V]|_c \delta V. \tag{III.48}$$

Expanding the first term in (III.48), we obtain

$$\begin{aligned}\frac{\partial}{\partial P_E}[\mathcal{M}_B^A \mathcal{R}V]|_c &= \frac{\partial \mathcal{M}}{\partial P_E}|_c {}_C^A \mathcal{R}V_C, \\ &= \left[\begin{array}{c|c|c} \frac{\partial \mathcal{M}}{\partial P_{E_1}}|_c & \frac{\partial \mathcal{M}}{\partial P_{E_2}}|_c & \frac{\partial \mathcal{M}}{\partial P_{E_3}}|_c \end{array}\right] {}_C^A \mathcal{R}V_C, \\ &= \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 0 & \frac{\partial (1-y\kappa)^{-1}}{\partial y}|_c & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial -z\tau(1-y\kappa)^{-1}}{\partial y}|_c & 0 & 0 & \frac{\partial -z\tau(1-y\kappa)^{-1}}{\partial z}|_c & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial -y\tau(1-y\kappa)^{-1}}{\partial y}|_c & 0 & 0 & 0 & 0 & 0 \end{array}\right] {}_C^A \mathcal{R}V_C,\end{aligned}$$

$$= \left[\begin{array}{ccc||ccc||ccc} 0 & 0 & 0 & \kappa({}^A_C\mathcal{R}V_C)_x & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\tau({}^A_C\mathcal{R}V_C)_x & 0 & 0 \\ 0 & 0 & 0 & -\tau({}^A_C\mathcal{R}V_C)_x & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$= ({}^A_C\mathcal{R}V_C)_x \left[\begin{array}{ccc} 0 & \kappa & 0 \\ 0 & 0 & -\tau \\ 0 & -\tau & 0 \end{array}\right], \qquad \text{(III.49)}$$

where the subscript c implies that the preceding gradient is being evaluated along the trimming trajectory. The subscript x implies the x-component of the vector, and

$$P_E = \left[\begin{array}{ccc} P_{E_1} & P_{E_2} & P_{E_2} \end{array}\right]^T.$$

Now, from the definition of Λ_E, it follows that

$$\frac{\partial \Lambda}{\partial \Lambda_E} = \mathcal{Q}. \qquad \text{(III.50)}$$

Next, using equation III.50 and Identity 1, we get

$$\begin{aligned} \frac{\partial}{\partial \Lambda_E}[\mathcal{M}{}^A_B\mathcal{R}V]|_c &= (\mathcal{M}\frac{\partial}{\partial \Lambda_E}[{}^A_B\mathcal{R}V])|_c, \\ &= (\frac{\partial}{\partial \Lambda_E}[{}^A_I\mathcal{R}{}^I_B\mathcal{R}V])|_c, \\ &= ({}^A_I\mathcal{R}\frac{\partial}{\partial \Lambda}[{}^I_B\mathcal{R}V]\frac{\partial \Lambda}{\partial \Lambda_E}])|_c, \\ &= -({}^A_I\mathcal{R}{}^I_B\mathcal{R}\mathcal{S}(V)\mathcal{Q}^{-1}\mathcal{Q})|_c, \\ &= -{}^A_C\mathcal{R}\mathcal{S}(V_C), \qquad \text{(III.51)} \end{aligned}$$

and

$$\frac{\partial}{\partial V}[\mathcal{M}{}^A_B\mathcal{R}V]|_c = {}^A_C\mathcal{R}. \qquad \text{(III.52)}$$

By combining equations (III.49 - III.52), we obtain

$$\frac{d}{dt}\delta P_E = ({}^A_C\mathcal{R}V_C)_x \left[\begin{array}{ccc} 0 & \kappa & 0 \\ 0 & 0 & -\tau \\ 0 & -\tau & 0 \end{array}\right] \delta P_E \qquad \text{(III.53)}$$

$$- {}^A_C\mathcal{R}\mathcal{S}(V_C)\delta\Lambda_E + {}^A_C\mathcal{R}\delta V_E. \qquad \text{(III.54)}$$

The final term, $\frac{d}{dt}\delta\lambda_E$, is obtained by observing that, along the trajectories $P_C \in \mathcal{E}$, $\frac{d}{dt}\mathcal{Q}^{-1} = 0$. Using simple algebra, we obtain

$$\frac{d}{dt}\delta\Lambda_E = \delta\Omega_E - \frac{\partial}{\partial\Lambda}(\mathcal{Q}^{-1}\frac{d}{dt}\Lambda_C)\frac{\partial\Lambda}{\partial\Lambda_E}|_c\delta\Lambda_E. \tag{III.55}$$

Now, since $\frac{d}{dt}\Lambda_C = [0 \;\; 0 \;\; \dot{\psi}_c]^T$ and from the definition of $\mathcal{Q}^{-1}$ (III.3) and ${}^B_I\mathcal{R}$ (III.2), it follows that

$$\begin{aligned}\mathcal{Q}^{-1}\frac{d}{dt}\Lambda_C &= \Omega_C, \\ &= {}^B_I\mathcal{R}\frac{d}{dt}\Lambda_C.\end{aligned} \tag{III.56}$$

Finally, using Identity 1 and equations III.50 and III.56, we obtain:

$$\begin{aligned}\frac{\partial}{\partial\Lambda}(\mathcal{Q}^{-1}\frac{d}{dt}\Lambda_C)\frac{\partial\Lambda}{\partial\Lambda_E}|_c &= \frac{\partial}{\partial\Lambda}({}^B_I\mathcal{R}\frac{d}{dt}\Lambda_C)\mathcal{Q}|_c, \\ &= \mathcal{S}({}^B_I\mathcal{R}\frac{d}{dt}\Lambda_C)\mathcal{Q}^{-1}\mathcal{Q}|_c = \mathcal{S}(\Omega_C).\end{aligned} \tag{III.57}$$

The desired expression for $\frac{d}{dt}\delta\Lambda_E$ now follows from expressions III.55 and III.57 as

$$\frac{d}{dt}\delta\Lambda_E = \delta\Omega_E - \mathcal{S}(\Omega_C)\delta\Lambda_E. \tag{III.58}$$

Summarizing, the linearization of equations III.34 along any trajectory $P_C \in \mathcal{E}$ is *time-invariant* and given by

$$\mathcal{G}_l\,(\eta_c) = \left\{ \begin{array}{rcl} \frac{d}{dt}\delta V_E &=& \mathcal{A}^V_V\delta V_E + \mathcal{A}^V_\Omega\delta\Omega_E + \mathcal{A}^V_\Lambda\delta\Lambda_E + \mathcal{B}_V\delta U, \\ \frac{d}{dt}\delta\Omega_E &=& \mathcal{A}^\Omega_V\delta V_E + \mathcal{A}^\Omega_\Omega\delta\Omega_E + \mathcal{A}^\Omega_\Lambda\delta\Lambda_E + \mathcal{B}_\Omega\delta U, \\ \frac{d}{dt}\delta P &=& ({}^A_C\mathcal{R}V_C)_x \begin{bmatrix} 0 & \kappa & 0 \\ 0 & 0 & -\tau \\ 0 & -\tau & 0 \end{bmatrix}\delta P_E, \\ &-& {}^A_C\mathcal{R}\mathcal{S}(V_C)\delta\Lambda_E + {}^A_C\mathcal{R}\delta V_E, \\ \frac{d}{dt}\delta\Lambda_E &=& \delta\Omega_E - \mathcal{S}(\Omega_C)\delta\Lambda_E, \end{array} \right. \tag{III.59}$$

where $({}^A_C\mathcal{R}V_C)_x$ is the x-component of the vector ${}^A_C\mathcal{R}V_C$. In the next section, the symbol $\mathcal{G}_l$ will denote the set of all linear plants $\mathcal{G}_l\,(\eta_c)$ associated with the set $\mathcal{E}$.

D. TRAJECTORY TRACKING CONTROL SYSTEM DESIGN AND IMPLEMENTATION

1. Linear Controller Design

In the previous section, we have shown that the linearization of the nonlinear system $\mathcal{G}_E$ about any trajectory in $\mathcal{E}$ results in a time-invariant plant $\mathcal{G}_l\,(\eta_c)$. Therefore, associated with the set $\mathcal{E}$ there is a family of linear plants, $\mathcal{G}_l$. These can now be used to synthesize a tracking (possibly gain-scheduled) controller $\mathcal{C}$, which is designed to operate over all the trajectories in $\mathcal{E}$.

A common approach to the development of such a controller $\mathcal{C}$ requires designing a family of linear controllers for a *finite* number of linear plants in $\mathcal{G}_l$, and then interpolating between these controllers to achieve adequate performance for all the linearized plants in $\mathcal{G}_l$. During real time operation, the controller parameters are updated as functions of a *gain-scheduling variable* $q = h(V, \Omega, \Lambda, P, U, \eta_c)$, where h is a C^1 function. For example, a typical gain-scheduling variable, for the case of aircraft, is dynamic pressure. In that case, $q = \frac{1}{2}\rho\|V\|^2$, where ρ represents air density and is itself a function of aircraft height.

In what follows, we restrict ourselves to the idealized case where the description of each controller for each plant $\mathcal{G}_l\,(\eta_c)$ is available [Ref. 39]. Therefore, we assume that the first design step produces the set

$$\mathcal{C}_l \;:= \{\mathcal{C}_l\,(q_c), q_c = h(V_C, \Omega_C, \Lambda_C, P_C, U_c, \eta_c)\},$$

given by (see Figure 33):

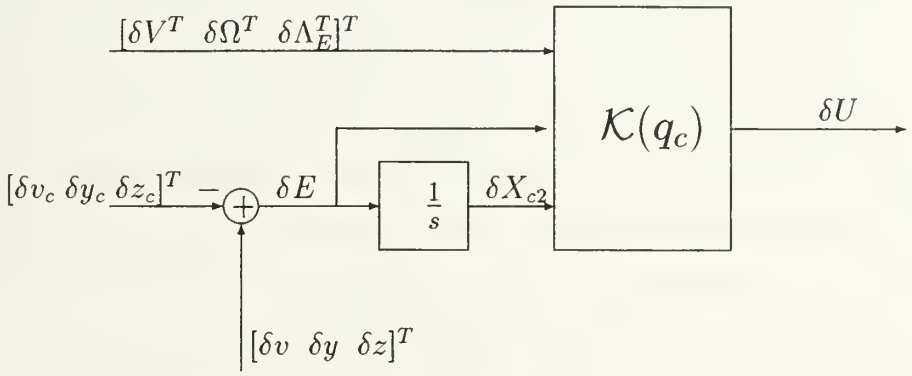
$$\mathcal{C}_l\,(q_c) = \left\{ \begin{array}{rcl} \delta E &=& [\delta v \;\; \delta y \;\; \delta z]^T - [\delta v_c \;\; \delta y_c \;\; \delta z_c]^T, \\ \frac{d}{dt}\delta X_{c1} &=& \mathcal{A}_{c1}(q_c)\delta X_{c1} + \mathcal{B}_{c1}(q_c)[\delta V^T \;\delta\Omega^T \;\delta\Lambda_E^T]^T \\ && +\mathcal{B}_{c2}(q_c)\delta X_{c2} + \mathcal{B}_{c3}(q_c)\delta E, \\ \frac{d}{dt}\delta X_{c2} &=& \delta E, \\ \delta U &=& \mathcal{C}_{c1}(q_c)\delta X_{c1} + \mathcal{D}_{c1}(q_c)[\delta V^T \;\delta\Omega^T \;\delta\Lambda_E^T]^T \\ && +\mathcal{D}_{c2}(q_c)\delta X_{c2} + \mathcal{D}_{c3}(q_c)\delta E, \end{array} \right. \qquad \text{(III.60)}$$

where $\delta X_{c1} \in R^{n_c}$, $\delta X_{c2} \in R^m$ and $m = \dim(U)$. The vector δX_{c2} represents the integrator states of the controller $\mathcal{C}_l\ (q_c)$. The vector δX_{c1} represents the remaining states of the controller $\mathcal{C}_l\ (q_c)$. The velocity and positon error is

$$\begin{bmatrix} \delta v \\ \delta y \\ \delta z \end{bmatrix}^T = \begin{bmatrix} \frac{V_C^T}{\|V_C\|} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta V \\ \delta \Omega \\ \delta \Lambda_E \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \delta P_E,$$

$$=: \mathcal{C}_1 \begin{bmatrix} \delta V \\ \delta \Omega \\ \delta \Lambda_E \end{bmatrix} + \mathcal{C}_2 \delta P_E, \qquad \text{(III.61)}$$

where δv_c, δy_c and δz_c are introduced to determine how fast the error states, δv, δy, and δz, go to zero. We further assume that the parameters of the controller are C^1 functions of q_c.



$$\mathcal{K}(q_c) = \left[\begin{array}{c|cc} A_{c1}(q_c) & B_{c1}(q_c) & B_{c2}(q_c) \quad B_{c3}(q_c) \\ \hline C_{c1}(q_c) & D_{c1}(q_c) & D_{c2}(q_c) \quad D_{c3}(q_c) \end{array}\right]$$

Figure 33. Set of Linear Controllers

The structure of the controller $\mathcal{C}_l\ (q_c)$ has the following important feature. Suppose the closed-loop system consisting of $\mathcal{G}_l\ (q_c)$ and $\mathcal{C}_l\ (q_c)$ given by equations III.59 and III.60 is asymptotically stable. Then, for a given q_c, the controller $\mathcal{C}_l\ (q_c)$

will ensure zero steady-state error to a step input for the variables in δE. This includes errors in the vehicle's inertial velocity v, and in the deviations from P_C, z and y. Zero steady state errors are achieved by integrating δE. This structure is typical of tracking controllers, since they are designed to drive errors between step changes in reference commands and the corresponding plant outputs to zero in steady state. Notice that the block $\mathcal{K}(q_c)$ (see Figure 33) may itself contain additional integrators.

2. Gain Scheduled Controller Design

Next, the family of linear controllers $\mathcal{C}_l\ (q_c)$ must be implemented on the nonlinear plant $\mathcal{G}$ defined in Section B. This problem has been addressed in [Ref. 27] for the general class of nonlinear plants and for tracking controllers with the same structure as $\mathcal{C}_l\ (q_c)$. In [Ref. 27], the authors formulated a so-called *controller implementation problem* which will be repeated here for the problem at hand.

Let $\mathcal{T}(\mathcal{G}_l\ (q_c),\ \mathcal{C}_l\ (q_c))$ be the closed-loop linear system that results from connecting $\mathcal{C}_l\ (q_c)$ to $\mathcal{G}_l\ (q_c)$, and denote by $T(\mathcal{G}_l\ (q_c),\ \mathcal{C}_l\ (q_c))$ the corresponding matrix transfer function. Let $\mathcal{T}(\mathcal{G}\ ,\ \mathcal{C}\ )(q_c)$ be the nonlinear closed-loop system that consists of $\mathcal{C}$ and $\mathcal{G}$, and let $\mathcal{T}_l(\mathcal{G}\ ,\ \mathcal{C}\ )(q_c)$ denote its linearization about $P_C \in \mathcal{E}$, and denote by $T_l(\mathcal{G}\ ,\ \mathcal{C}\ )(q_c)$ the corresponding matrix transfer function. With this notation, the controller implementation problem applied to the integrated guidance and control problem considered in this chapter can be stated as follows:

Controller implementation problem: "Find a gain scheduled controller $\mathcal{C}$ such that for each trajectory $P_C \in \mathcal{E}$

1. the feedback systems $\mathcal{T}_l(\mathcal{G}\ ,\ \mathcal{C}\ )(q_c)$ and $\mathcal{T}(\mathcal{G}_l\ (q_c),\ \mathcal{C}_l\ (q_c))$ have the same closed-loop eigenvalues.
2. The closed-loop transfer functions $T_l(\mathcal{G}\ ,\ \mathcal{C}\ )(q_c)$ and $T(\mathcal{G}_l\ (q_c),\ \mathcal{C}_l\ (q_c))$ are "equal."

Given the set $\mathcal{C}_l$ of linear controllers for the set $\mathcal{G}_l$ of linearized plant models,

we propose the following structure for the gain scheduled controller $\mathcal{C}$ (see Figure 34):

$$\mathcal{C} := \left\{ \begin{array}{rcl} [0 \;\; y \;\; z]^T & = & {}^{A}_{I}\mathcal{R}(P - P_C(s_0)), \\ E & = & [v - v_c \;\; y - y_c \;\; z - z_c]^T, \\ \frac{d}{dt}X_{c1} & = & \mathcal{A}_{c1}(q)X_{c1} + \mathcal{B}_{c1}(q)[\frac{d}{dt}V^T \;\; \frac{d}{dt}\Omega^T \;\; (\Omega - \mathcal{Q}^{-1}\dot{\Lambda}_C)^T]^T \\ & & +\mathcal{B}_{c2}(q)E + \mathcal{B}_{c3}(q)\frac{d}{dt}E, \\ \frac{d}{dt}X_{c2} & = & \mathcal{C}_{c1}(q)X_{c1} + \mathcal{D}_{c1}(q)[\frac{d}{dt}V^T \;\; \frac{d}{dt}\Omega^T \;\; (\Omega - \mathcal{Q}^{-1}\dot{\Lambda}_C)^T]^T \\ & & +\mathcal{D}_{c2}(q)E + \mathcal{D}_{c3}(q)\frac{d}{dt}E, \\ U & = & X_{c2}. \end{array} \right. \qquad \text{(III.62)}$$

Recall, $P_C(s_0)$ is the projection of P onto the helix $P_C \in \mathcal{E}$. Comparison of Figure 33 and Figure 34 indicates that the structure of the gain scheduled controller is easily obtained from that of the linear controllers.

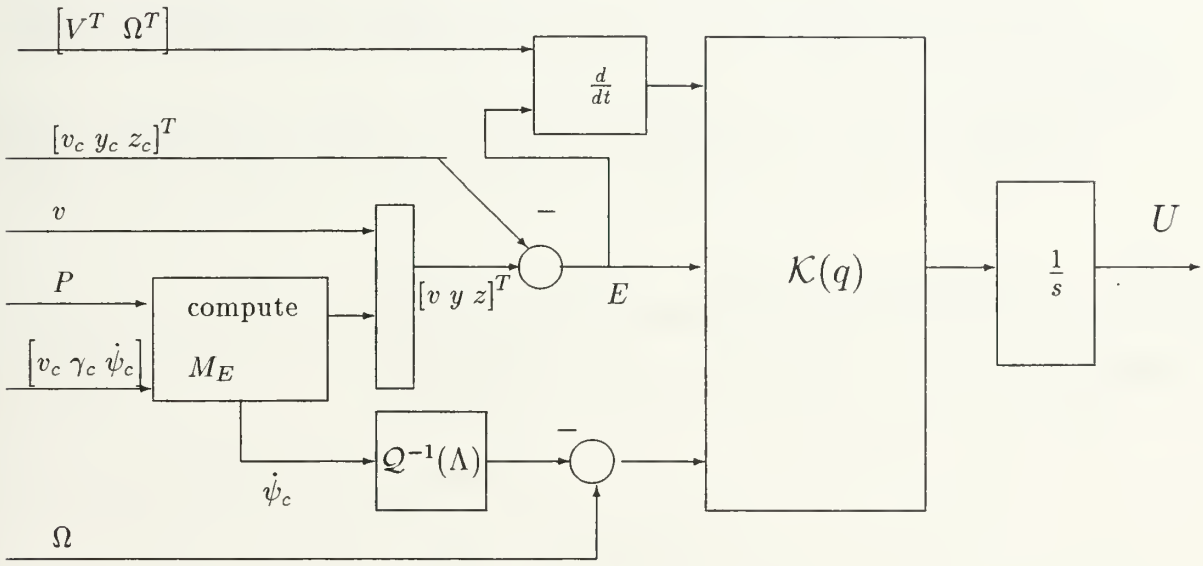


Figure 34. Gain Scheduled Controller

We now make the following assumptions:

*A*1. $\text{Dim}(X_{c2}) = \dim(U) = \dim(E)$

*A*2. The matrix

$$\begin{bmatrix} sI - \mathcal{A}_{c1}(q) & \mathcal{B}_{c2}(q) \\ -\mathcal{C}_{c1}(q) & \mathcal{D}_{c2}(q) \end{bmatrix}$$

has full rank at $s = 0$ for each $P_C \in \mathcal{E}$.

$A3$. The matrix pair $(\mathcal{A}_{c1}, \mathcal{C}_{c1})$ is observable.

Assumption $A1$ implies that the number of integrators is equal to the number of control inputs. This is necessary if the controller is to provide independent control of the errors E using the control inputs U. Assumption $A2$ implies that the realization $(\mathcal{A}_{c1}, \mathcal{B}_{c2}, \mathcal{C}_{c1}, \mathcal{D}_{c2})$ has no transmission zeroes at the origin. Finally, assumption $A3$ guarantees that the state X_{c1} is zero along the trajectories in $\mathcal{E}$.

The main result of this section is stated next.

Theorem D.1 *Suppose assumptions A1, A2 hold. Then the gain scheduled controller $\mathcal{C}$ given by equations III.62 solves the controller implementation problem, i.e., for each $P_C \in \mathcal{E}$ the following properties hold:*

1. The feedback systems $\mathcal{T}_l(\mathcal{G}\ ,\ \mathcal{C}\ )(q_c)$ and $\mathcal{T}(\mathcal{G}_l\ (q_c),\ \mathcal{C}_l\ (q_c))$ have the same closed-loop eigenvalues.
2. The closed-loop matrix transfer functions $T_l(\mathcal{G}\ ,\ \mathcal{C}\ )(q_c)(s)$ and $T(\mathcal{G}_l\ (q_c),\ \mathcal{C}_l\ (q_c))(s)$ are equal.

Proof: In the proof, we set the controller matrices $\mathcal{D}_{c1}$, $\mathcal{D}_{c3}$ to zero. This does not change the results but considerably simplifies the algebra.

Let $P_C \in \mathcal{E}$ be given. Let

$$\mathcal{A}_1 := \begin{bmatrix} \mathcal{A}_V^V & \mathcal{A}_\Omega^V & \mathcal{A}_\Lambda^V \\ \mathcal{A}_V^\Omega & \mathcal{A}_\Omega^\Omega & \mathcal{A}_\Lambda^\Omega \\ 0 & I & -\mathcal{S}(\Omega_C) \end{bmatrix},$$

$$\mathcal{A}_2 := \begin{bmatrix} {}^A_C\mathcal{R} & 0 & -{}^A_C\mathcal{R}\mathcal{S}(V_C), \end{bmatrix},$$

$$\mathcal{A}_3 := ({}^A_C\mathcal{R}V_C)_x \begin{bmatrix} 0 & \kappa & 0 \\ 0 & 0 & -\tau \\ 0 & -\tau & 0 \end{bmatrix},$$

$$\mathcal{B} := \begin{bmatrix} \mathcal{B}_V \\ \mathcal{B}_\Omega \\ 0 \end{bmatrix},$$

$$({}^A_C\mathcal{R}V_C)_x := x\text{-component of } {}^A_C\mathcal{R}V_C$$

$$\delta\Upsilon := \begin{bmatrix} \delta V_E \\ \delta\Omega_E \\ \delta\Lambda_E \end{bmatrix},$$

Then, (III.59) is expressed as

$$\mathcal{G}_l(\eta_c) = \begin{cases} \frac{d}{dt}\delta\Upsilon &= \mathcal{A}_1\delta\Upsilon + \mathcal{B}\delta U, \\ \frac{d}{dt}\delta P &= \mathcal{A}_2\delta\Upsilon + \mathcal{A}_3\delta P. \end{cases} \tag{III.63}$$

The error signal is expressed as

$$\delta E = \mathcal{C}_1\delta\Upsilon + \mathcal{C}_2\delta P, \tag{III.64}$$

which, when substituted into the linear controller (III.60), results in

$$\mathcal{C}_l(q_c) = \begin{cases} \dot{\delta}X_{c1} &= \mathcal{A}_{c1}(q_c)\delta X_{c1} + \mathcal{B}_{c2}\delta X_{c2} + [\mathcal{B}_{c1}(q_c) + \mathcal{B}_{c3}(q_c)\mathcal{C}_1]\,\delta\Upsilon \\ &\quad + \mathcal{B}_{c3}(q_c)\mathcal{C}_2\delta P, \\ \frac{d}{dt}\delta X_{c2} &= \mathcal{C}_1\delta\Upsilon + \mathcal{C}_2\delta P, \\ \delta U &= \mathcal{C}_{c1}(q_c)\delta X_{c1}\mathcal{D}_{c2}(q_c)\delta X_{c2}. \end{cases} \tag{III.65}$$

Consider the feedback interconnection of the linear plant (III.63) and linear controller (III.65). The state matrix F of this feedback system has the following form:

$$F := \begin{bmatrix} \mathcal{A}_1 & \mathcal{B}\mathcal{D}_{c3}\mathcal{C}_1 & \mathcal{B}\mathcal{C}_{c1} & \mathcal{B}\mathcal{D}_{c2} \\ \mathcal{A}_2 & \mathcal{A}_3 & 0 & 0 \\ \mathcal{B}_{c1} + \mathcal{B}_{c3}\mathcal{C}_1 & \mathcal{B}_{c3}\mathcal{C}_2 & \mathcal{A}_{c1} & \mathcal{B}_{c2} \\ \mathcal{C}_1 & \mathcal{C}_2 & 0 & 0 \end{bmatrix}. \tag{III.66}$$

Next, we linearize the feedback interconnection of the plant $\mathcal{G}$ and the controller $\mathcal{C}$. However, in order to do that, first we must determine the values of the

controller states X_{c1} and X_{c2} along the trajectory $P_C \in \mathcal{E}$. From equation III.62, we obtain:

$$\begin{aligned}
\frac{d}{dt}X_{c1_c} &= \mathcal{A}_{c1}(q_c)X_{c1_c} + \mathcal{B}_{c1}(q_c)[\frac{d}{dt}V_C^T \;\; \frac{d}{dt}\Omega_C^T \;\; (\Omega_C - \mathcal{Q}_C^{-1}\dot{\Lambda}_C)^T]^T \\
&\quad + \mathcal{B}_{c2}(q_c)E + \mathcal{B}_{c3}(q_c)\frac{d}{dt}E, \\
\frac{d}{dt}X_{c2_c} &= \mathcal{C}_{c1}(q_c)X_{c1_c} + \mathcal{D}_{c2}(q_c)E, \\
U_c &= X_{c2_c}.
\end{aligned} \tag{III.67}$$

Notice, since along $P_C \in \mathcal{E}$:

$$\begin{aligned}
E &= 0, \\
V_C &= \text{constant}, \\
\Omega_C &= \text{constant}, \\
\Omega_C - \mathcal{Q}_C^{-1}\dot{\Lambda}_C &= 0, \\
X_{c2_c} &= U_c, \text{ a constant},
\end{aligned} \tag{III.68}$$

we get

$$\begin{aligned}
\frac{d}{dt}X_{c1_c} &= \mathcal{A}_{c1}X_{c1_c}, \\
0 &= \mathcal{C}_{c1}X_{c1_c}.
\end{aligned}$$

Now, using Assumption $A3$, we conclude that

$$X_{c1_c} = 0. \tag{III.69}$$

In order to compute the linearization of the feedback interconnection of $\mathcal{G}$ and $\mathcal{C}$ $(\mathcal{T}(\mathcal{G}, \mathcal{C}))$ along $P_C \in \mathcal{E}$, first observe that $\mathcal{F}(\mathcal{G}, \mathcal{C})$ is equal to the feedback interconnection of $\mathcal{G}_E$ and the system consisting of $\mathcal{K}(q)$ and the integrator X_{c2}. The linearization of $\mathcal{G}_E$ along $P_C \in \mathcal{E}$ is given by (III.59). The linearization of the system consisting of $\mathcal{K}(q)$ and X_{c2} can be obtained using the trimming values of X_{c1} and X_{c2}.

Define

$$\begin{aligned}\frac{d}{dt}\theta &:= \left[\begin{array}{ccc} \frac{d}{dt}V_E^T & \frac{d}{dt}\Omega_E^T & (\Omega - \mathcal{Q}^{-1}\dot{\Lambda}_C)^T \end{array}\right]^T, \\ \varpi^T &:= \left[\begin{array}{ccccccc} X_{c1}^T & X_{c2}^T & \frac{d}{dt}\theta & E^T & \frac{d}{dt}E^T & q_c \end{array}\right]^T,\end{aligned}$$

then, along P_C

$$\varpi_0^T = \left[\begin{array}{cccccc} 0 & U_c^T & 0 & 0 & 0 & q_{c_0} \end{array}\right]^T, \tag{III.70}$$

and the linearization of (III.67) can be written as

$$\begin{aligned}\frac{d}{dt}\delta X_{c1_c} &= \frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1_c})}{\partial\varpi}|_{\varpi_0} + \frac{\partial(\mathcal{B}_{c1}(q_c)\frac{d}{dt}\theta}{\partial\varpi}|_{\varpi_0} + \frac{\partial(\mathcal{B}_{c2}(q_c)E)}{\partial\varpi}|_{\varpi_0} \\ &\quad + \frac{\partial(\mathcal{B}_{c3}(q_c)\frac{d}{dt}E)}{\partial\varpi}|_{\varpi_0}, \\ \frac{d}{dt}\delta X_{c2_c} &= \frac{\partial(\mathcal{C}_{c1}(q_c)X_{c1_c})}{\partial\varpi}|_{\varpi_0} + \frac{\partial(\mathcal{D}_{c2}(q_c)E)}{\partial\varpi}|_{\varpi_0} \\ \delta U_c &= \frac{\partial(X_{c2_c})}{\partial\varpi}|_{\varpi_0}.\end{aligned} \tag{III.71}$$

Consider the expansion of the first term in (III.71),

$$\begin{aligned}\frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1_c})}{\partial\varpi}|_{\varpi_0} &= \frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1_c})}{\partial X_{c1}}|_{\varpi_0} + \frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1})}{\partial X_{c2}}|\varpi_0 + \frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1_c})}{\partial(\frac{d}{dt}\theta)}|\varpi_0 \\ &\quad + \frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1_c})}{\partial E}|\varpi_0 + \frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1_c})}{\partial\dot{E}}|\varpi_0 + \frac{\partial(\mathcal{A}_{c1}(q_c)X_{c1_c})}{\partial q_c}|_{\varpi_0}, \\ &= \mathcal{A}_{c1}(q_{c_0})\delta X_{c1_c}.\end{aligned} \tag{III.72}$$

Since X_{c1_c} is zero along the trajectory, their are no extra terms due to the scheduling parameter, q_c. Similar results are obtained for the terms $\mathcal{B}_{c1}(q_c)$, $\mathcal{C}_{c1}(q_c)$, and $\mathcal{D}_{c2}(q_c)$. Noting that

$$\delta\frac{d}{dt}\theta^T = \left[\begin{array}{ccc} \frac{d}{dt}\delta V_E^T & \frac{d}{dt}\delta\Omega_E^T & \frac{d}{dt}\delta\Lambda_E^T \end{array}\right]^T,$$

the linearization of (III.67) has the following form:

$$\begin{aligned}\frac{d}{dt}\delta X_{c1_c} &= \mathcal{A}_{c1}\delta X_{c1_c} + \mathcal{B}_{c1}[\frac{d}{dt}\delta V_E^T \;\; \frac{d}{dt}\delta\Omega_E^T \;\; \frac{d}{dt}\delta\Lambda_E^T]^T + \mathcal{B}_{c2}\delta E + \mathcal{B}_{c3}\frac{d}{dt}\delta E, \\ \frac{d}{dt}\delta X_{c2} &= \mathcal{C}_{c1}\delta X_{c1_c} + \mathcal{D}_{c2}\delta E, \\ \delta U &= \delta X_{c2_c}.\end{aligned} \tag{III.73}$$

The state matrix M of $\mathcal{T}_l(\mathcal{G}\ ,\ \mathcal{C}\ )$ is calculated as

$$M = \left[\begin{array}{cc|c|cc} \mathcal{A}_1 & 0 & 0 & \multicolumn{2}{c}{\mathcal{B}} \\ \mathcal{A}_2 & \mathcal{A}_3 & 0 & \multicolumn{2}{c}{0} \\ \hline \multicolumn{2}{c|}{\begin{bmatrix} \mathcal{B}_{c1} & \mathcal{B}_{c3} \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix} \begin{bmatrix} \mathcal{A}_1 & 0 \\ \mathcal{A}_2 & \mathcal{A}_3 \end{bmatrix} + \mathcal{B}_{c2} \begin{bmatrix} \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix}} & \mathcal{A}_{c1} & \multicolumn{2}{c}{\begin{bmatrix} \mathcal{B}_{c1} & \mathcal{B}_{c3} \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix} \begin{bmatrix} B \\ 0 \end{bmatrix}} \\ \hline \multicolumn{2}{c|}{\mathcal{D}_{c2} \begin{bmatrix} \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix}} & \mathcal{C}_{c1} & \multicolumn{2}{c}{0} \end{array}\right].$$

The proof of the first part of the theorem can now be shown by following the proof of Theorem 4.1 in [Ref. 27]. From Assumption A2, it follows that the matrix

$$\begin{bmatrix} \mathcal{A}_{c1} & \mathcal{B}_{c2} \\ \mathcal{C}_{c1} & \mathcal{D}_{c2} \end{bmatrix}$$

is invertible. Therefore, we will use this to define a very useful similarity transformation to show that F and M have the same the eigenvalues. Let

$$\begin{bmatrix} \mathcal{A}_{c1} & \mathcal{B}_{c2} \\ \mathcal{C}_{c1} & \mathcal{D}_{c2} \end{bmatrix}^{-1} =: \begin{bmatrix} X & Y \\ Z & W \end{bmatrix}, \tag{III.74}$$

and set

$$T := \begin{bmatrix} I & 0 & 0 \\ -X \begin{bmatrix} \mathcal{B}_{c1} & \mathcal{B}_{c3} \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix} & X & Y \\ -Z \begin{bmatrix} \mathcal{B}_{c1} & \mathcal{B}_{c3} \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix} & Z & W \end{bmatrix}. \tag{III.75}$$

Expression III.74 implies

$$\begin{aligned} \mathcal{A}_{c1}X + \mathcal{B}_{c2} &= I, \\ \mathcal{C}_{c1}Y + \mathcal{D}_{c2} &= I, \\ \mathcal{A}_{c1}Y + \mathcal{B}_{c2} &= 0, \\ \mathcal{C}_{c1}X + \mathcal{D}_{c2} &= 0. \end{aligned} \tag{III.76}$$

A simple computation using (III.76) and (III.75) verifies that

$$T^{-1} := \begin{bmatrix} & I & & 0 & 0 \\ \begin{bmatrix} \mathcal{B}_{c1} & \mathcal{B}_{c3} \end{bmatrix} & & \begin{bmatrix} I & 0 \\ \mathcal{C}_1 & \mathcal{C}_2 \end{bmatrix} & X & Y \\ & 0 & & Z & W \end{bmatrix}, \tag{III.77}$$

and that $F = TMT^{-1}$. Thus, F and M have the same the eigenvalues.

In order to prove the second part of the theorem, it will suffice to show that the linear controllers (III.73) and (III.60) have the same transfer function from

$$E = \begin{bmatrix} \delta v & \delta y & \delta z \end{bmatrix}^T - \begin{bmatrix} \delta v_c & \delta y_c & \delta z_c \end{bmatrix}^T,$$

and

$$\theta = \begin{bmatrix} \delta V^T & \delta \Omega^T & \delta \Lambda^T \end{bmatrix}^T, \tag{III.78}$$

to δU. A direct calculation of the Laplace transforms show that

$$\begin{aligned} U(s) = {} & \mathcal{C}_{c1}(sI - \mathcal{A}_{c1})^{-1} \left\{ \mathcal{B}_{c1} \begin{bmatrix} \delta V^T(s) & \delta \Omega^T(s) & \delta \Lambda^T(s) \end{bmatrix}^T \right. \\ & \left. + \frac{\mathcal{B}_{c2} E(s)}{s} + \mathcal{B}_{c3} E(s) \right\} + \mathcal{D}_{c1} \begin{bmatrix} \delta V^T(s) & \delta \Omega^T(s) & \delta \Lambda^T(s) \end{bmatrix}^T \\ & + \frac{\mathcal{D}_{c2} E(s)}{s} + \mathcal{D}_{c3} E(s), \end{aligned} \tag{III.79}$$

for both controllers. ■

Thus, the eigenvalues of the linearizations along each trajectory in $\mathcal{E}$ are preserved. Furthermore, the input-output behavior of the linearized operators is preserved in a well-defined sense. The reader is referred to [Ref. 27] for a complete discussion on approximations to this method that avoid using pure differentiation.

3. Implementation Procedure

The Theorem D.1 can be used as follows: first, determine the dynamics of the vehicle $\mathcal{G}$ and the set of trimming trajectories $\mathcal{E}$ the vehicle is required to track.

This set is parameterized by the range of trimming velocities (v_c), desired flight path angles (γ_c), and desired heading rates ($\dot{\psi}_c$). Recall, these variable constitute a gain-scheduling vector $q_c = [v_c \quad \gamma_c \quad \dot{\psi}_c]^T$. Next, rewrite the vehicle dynamics using generalized error coordinates V_E, Ω_E, Λ_E, and P_E to obtain the vehicle's error dynamics $\mathcal{G}_E$. Linearize $\mathcal{G}_E$ about a finite number of trajectories in $\mathcal{E}$. Use these linear models to design a finite number (k) of trajectory tracking controllers $\{\mathcal{C}_{l\ i}, i = 1, k\}$. Gain schedule $\{\mathcal{C}_{l\ i}, i = 1, k\}$ utilizing a favorite interpolation or gain-scheduling technique to obtain the linear gain-scheduled controller $\mathcal{C}_l\ (q_c)$. Now, implement this controller on the nonlinear plant $\mathcal{G}$ according to the expression III.62.

It is worth emphasizing the following important properties of the controller $\mathcal{C}$:

- The result in Theorem D.1 holds for all trajectories in $\mathcal{E}$.
- The structure of the controller $\mathcal{C}$ is easily obtained from that of the linear controllers.
- Since all the closed-loop transfer functions of the local linearizations are preserved, at the level of local linear analysis, the controller does not introduce any additional noise amplification despite the presence of a differentiation operator.
- Along trajectories $P_C \in \mathcal{E}$, $X_{c2_c} = U_c$ and $X_{c1_c} = 0$. Therefore, the trimming values of the control inputs are naturally provided by the integrator block with state X_{c2_c}.
- The integrators X_{c2} are directly at the input of the plant, which makes it straightforward to implement anti-windup schemes. This becomes necessary in applications where the input U is hard limited due to actuator saturation, for example.
- The inputs to the controller, P_C and $\dot{\Lambda}_C$, can be computed directly from the vector q_c.
- The trim values V_C, Ω_C, and U_c are not required in the controller implementation.
- Along the trajectories $P_C \in \mathcal{E}$, the controller guarantees that the steady state value of error vector E is zero, which follows from the fact that the controller solves the controller implementation problem. This is in sharp contrast to standard LOS guidance schemes.

E. EXAMPLE

In this section we apply the methodology developed in Section D to the design and implementation of an integrated guidance and control system for a fixed wing unmanned air vehicle. The vehicle's stability and control derivatives are the subject of a section of chapter IV. Using the notation in Section B, the *Frog* dynamics have the following form:

$$\mathcal{G} = \begin{cases} \frac{d}{dt} V &= \mathcal{F}_V(V,\Omega,\Lambda) + \mathcal{I}_V(V,\Omega)\mathcal{H}(V,\Omega,U), \\ \frac{d}{dt} \Omega &= \mathcal{F}_\Omega(V,\Omega,\Lambda) + \mathcal{I}_\Omega(V,\Omega)\mathcal{H}(V,\Omega,U), \\ \frac{d}{dt} P &= {}^I_B\mathcal{R} V, \\ \frac{d}{dt} \Lambda &= \mathcal{Q}\,\Omega. \end{cases} \tag{III.80}$$

Following the development in Section B, the set of trimming trajectories $\mathcal{E}$ for the vehicle is defined as follows:

$$\mathcal{E} := \left\{ \begin{bmatrix} P_C \\ \Lambda_C \end{bmatrix} \; : \; \begin{array}{l} \frac{d}{dt} P_C = {}^I_C\mathcal{R} V_C, \\ \frac{d}{dt} \Lambda_C = \mathcal{Q}_C\, \Omega_C, \\ \mathcal{F}_V(V_C,\Omega_C,\Lambda_C) + \mathcal{I}_V(V_C,\Omega_C)\mathcal{H}(V_C,\Omega_C,U_c) = 0, \\ \mathcal{F}_\Omega(V_C,\Omega_C,\Lambda_C) + \mathcal{I}_\Omega(V_C,\Omega_C)\mathcal{H}(V_C,\Omega_C,U_c) = 0, \end{array} \right\} \tag{III.81}$$

where P_C and Λ_C can be computed using the scheduling vector $q = [v_c \;\; \gamma_c \;\; \dot{\psi}_c]^T$. Now, given $[v_c \;\; \gamma_c \;\; \dot{\psi}_c]^T$ and $\beta_c = 0$, we can solve for V_C, Ω_C, U_c, and Λ_C:

$$\begin{aligned} &\mathcal{F}_V(V_C,\Omega_C,\Lambda_C) + \mathcal{I}_V(V_C,\Omega_C)\mathcal{H}(V_C,\Omega_C,U_c) = 0, \\ &\mathcal{F}_\Omega(V_C,\Omega_C,\Lambda_C) + \mathcal{I}_\Omega(V_C,\Omega_C)\mathcal{H}(V_C,\Omega_C,U_c) = 0, \\ &\mathcal{Q}_C^{-1}\Omega_C - \dot{\Lambda}_C = 0 \\ &\|V_C\| - v_c = 0, \\ &\beta_c - \sin^{-1}\frac{v_c}{V_t} = 0, \\ &\gamma_c - [0 \;\; 1 \;\; 0]\, arg({}^W_B\mathcal{R}^B_I\mathcal{R}) = 0, \end{aligned} \tag{III.82}$$

where the *arg* function extracts the angles X from the rotation matrix $\mathcal{R}(X)$: $X = arg(\mathcal{R}(X))$.

Using the solution to equations III.82, the linear model for the vehicle represented by equations III.59 was obtained along a trajectory characterized by the velocity of 73 feet per second, flight path angle of zero, and heading rate of 10 degrees per second. This model was used to design a linear trajectory tracking controller for the vehicle.

1. Design Requirements and Linear Controller Design

Design requirements for the linear trajectory tracking controller included

- **Zero Steady State Error:** Achieve zero steady state tracking errors of all trajectories in $\mathcal{E}$. Achieve zero steady state tracking error of indicated airspeed while on any trajectory in $\mathcal{E}$.
- **Bandwidth Requirements:** The command-loop bandwidth for each command channel should be no greater than 1 radian per second and no less than 1/10 radian per second; the control-loop bandwidth should not exceed 12 radians per second for the elevator, aileron and rudder loops, and 5 radians per second for the throttle loop. These numbers represent 50% of the corresponding actuator bandwidths, and shall ensure that the actuators are not driven beyond their linear operating range.
- **Closed Loop Damping and Stability Margins:** The dominant closed-loop eigenvalues should have a damping ratio of at least 0.5. Simultaneous gain and phase margins of 6db and 45 degrees in each control loop must be achieved.

The methodology selected for linear control system design was $\mathcal{H}_\infty$ synthesis [Ref. 12]. This method rests on a firm theoretical basis, and leads naturally to an interpretation of control design specifications in the frequency domain. Furthermore, it provides clear guidelines for the design of controllers so as to achieve robust performance in the presence of plant uncertainty. The basic steps in the controller-design procedure, including the development of the synthesis model, were done using the approach described in [Ref. 24]. This approach provides an intuitive and straightforward way for converting the design requirements into the weights for the

$\mathcal{H}_\infty$ synthesis model. Consider Figure 35. Here $\mathcal{C}_l$ is the controller to be designed, $\mathcal{G}_l$ is the linear model of the vehicle.

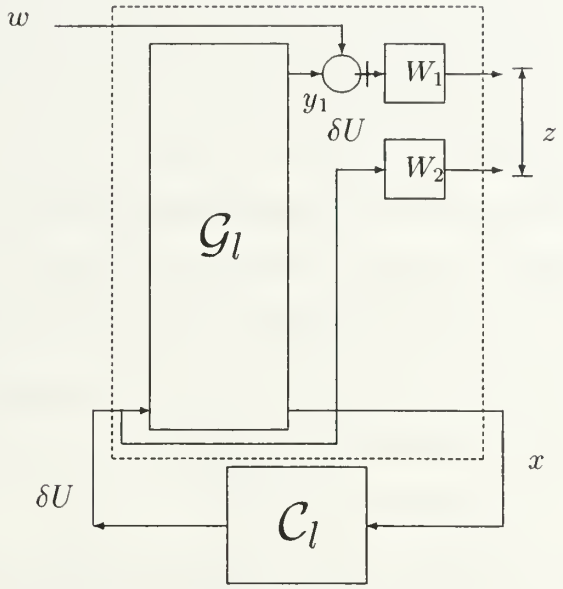


Figure 35. Synthesis Model

In Figure 35, the vector of exogenous inputs w represents the commanded inputs. The vector y_1 represents lateral and vertical displacement states of the linear model as well as the vehicle's velocity. The regulated output z includes the outputs of the weighting matrices W_1 and W_2. These matrices had the following form:

$$W_1 = \begin{bmatrix} \frac{c_1}{s} & 0 & 0 \\ 0 & \frac{c_2}{s} & 0 \\ 0 & 0 & \frac{c_3}{s} \end{bmatrix},$$

$$W_2 = \begin{bmatrix} c_4 & 0 & 0 \\ 0 & c_5 & 0 \\ 0 & 0 & c_6 \end{bmatrix}, \qquad \text{(III.83)}$$

where the constants c_i, $i = 1, 6$ were used as the design knobs adjusted to meet the closed-loop tracking, damping, control, and command loop bandwidth requirements.

Values of W_1 and W_2 were iterated on, and used to obtain a linear trajectory tracking controller:

$$\mathcal{C}_l\,(\bar{q},) := \begin{cases} \delta E & = [\delta v \;\; \delta y \;\; \delta z]^T - [\delta v_c \;\; \delta y_c \;\; \delta z_c]^T, \\ \frac{d}{dt}\delta X_{c1} & = \mathcal{A}_{c1}\delta X_{c1} + \mathcal{B}_{c1}\delta X_{c2}, \\ \frac{d}{dt}\delta X_{c2} & = \delta E, \\ \delta U & = \frac{\bar{q}_0}{\bar{q}}\{\mathcal{C}_{c1}\delta X_{c1} + \mathcal{D}_{c1}[\delta V^T \;\; \delta\Omega^T \;\; \delta\Lambda_E^T]^T + \mathcal{D}_{c2}\delta X_{c2} + \mathcal{D}_{c3}\delta E\}, \end{cases}$$

where the $\mathcal{H}_\infty$ state-feedback gain is $\mathcal{K} = [\mathcal{C}_{c1} \;\; \mathcal{D}_{c1} \;\; \mathcal{D}_{c2} \;\; \mathcal{D}_{c3}]$. The feedback system consisting of the plant $\mathcal{G}_l$ and the controller $\mathcal{C}_l$ was found to meet all the design specifications given earlier in this section. Since the control surface effectiveness is proportional to the dynamic pressure, the controller $\mathcal{C}_l$ was gain-scheduled on $\bar{q}$. The value $\bar{q}_0$ represents the nominal value of $\bar{q}$.

2. Implementation and Simulation Results

Using the formulae provided in Section D, the family of linear gain-scheduled controllers $\mathcal{C}_l\,(\bar{q})$ was implemented on the nonlinear plant $\mathcal{G}$ as follows:

$$\mathcal{C} := \begin{cases} [0 \;\; y \;\; z]' & =_I^A \mathcal{R}(P - P_C(s_0)), \\ E & = [v - v_c \;\; y \;\; z], \\ \dot{X}_{c1} & = \mathcal{A}_{c1}X_{c1} + \mathcal{B}_{c2}E, \\ \dot{X}_{c2} & = \frac{\bar{q}_0}{\bar{q}}\{\mathcal{C}_{c1}X_{c1} + \mathcal{D}_{c1}[\frac{d}{dt}V^T \;\; \frac{d}{dt}\Omega^T \;\; \Omega - \mathcal{Q}^{-1}(\Lambda)\frac{d}{dt}\Lambda_C]^T + \mathcal{D}_{c3}\frac{d}{dt}E + \mathcal{D}_{c2}E\}, \\ U & = X_{c2}. \end{cases}$$

We emphasize that the implementation equations for the controller $\mathcal{C}$ do not require the computation of Ω_C, U_c and V_C. Moreover, since $\frac{d}{dt}\Lambda_C = [0 \;\; 0 \;\; \dot{\psi}_c]^T$, the controller must only be provided with $\dot{\psi}_c$ and P_C when steering the aircraft along the trajectory. These are the critical advantages of the proposed methodology. The acceleration term $\frac{d}{dt}V$ can be computed using onboard sensors without resorting to differentiation. Therefore, the only term which could not be computed directly was $\frac{d}{dt}\Omega$. In this case, the differentiation operator $\frac{d}{dt}$ was replaced by a causal operator with the transfer function $\frac{s}{\tau s+1}$ [Ref. 27].

The trajectory tracking controller developed above was tested using a number of trajectories. One such trajectory consisted of a straight line transitioning into a helix shown in Figure 36. This trajectory is characterized by a cruise velocity of 73 feet per second. Initially, the trajectory is aligned with the inertial x-axis. After proceeding along the x axis for 3000 feet, the trajectory turns into a helix with a radius of 1000 feet and climb angle of 5 degrees. Consider Figure 37, which shows the time history of the position error, bank and pitch angles, and indicated airspeed along the trajectory. Clearly, the controller drives the vehicle along this trajectory with zero steady state position errors while maintaining 73 fps indicated airspeed.

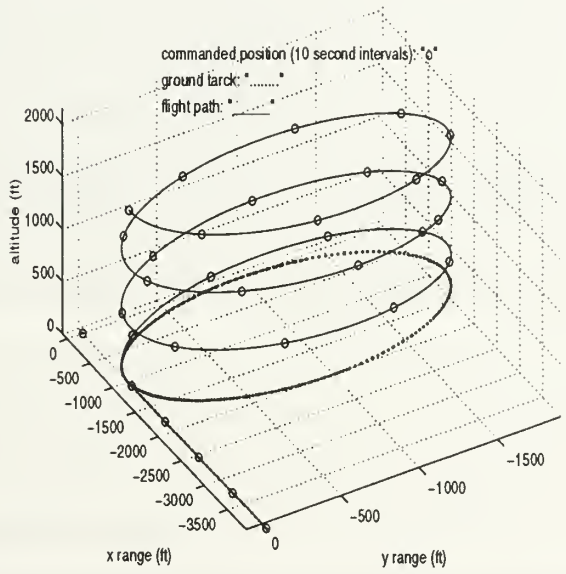


Figure 36. The trajectory tracked in simulation.

F. CONCLUSIONS

A new method was introduced for designing and implementing integrated guidance and control systems for autonomous vehicles. The starting point is a family of linear controllers with integral action designed for linearizations of the nonlinear equations of motion described in an appropriate state space. Based on this family, the method produces a gain scheduled controller that preserves the input-output proper-

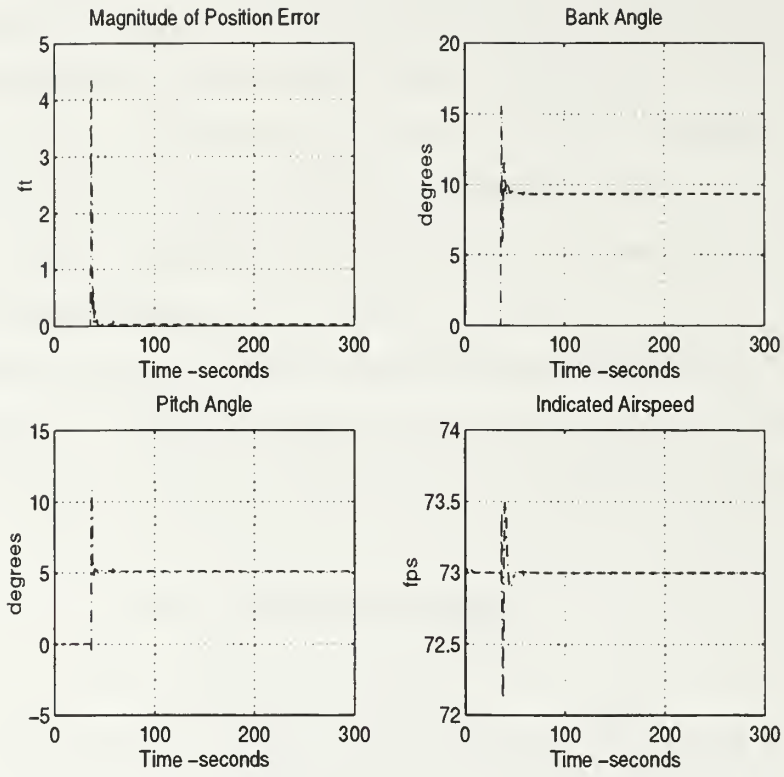


Figure 37. Time history of position errors, Euler angles and airspeed along the trajectory.

ties of the original linear closed-loop systems as well as the closed-loop eigenvalues. The key feature of the method is the ability to automatically reconfigure the control inputs of the vehicle to provide for proper control action as the body tracks an inertial trajectory in free space while maintaining constant airspeed. The method is simple to apply and leads to a nonlinear controller with a structure similar to that of the original linear design.

IV. RAPID PROTOTYPING SYSTEM FOR FLIGHT TEST OF AN UAV

A. INTRODUCTION

This chapter describes the development of a rapid prototyping system for flight testing of guidance, navigation, and control algorithms for unmanned air vehicles. The system affords a small team the ability to take a new concept in guidance, navigation, and control from initial conception to flight test. In order to do this, a number of engineering problems had to be overcome addressing a gamut of issues including weight, power, portability, risk, electronic interference, vibration, manpower, etc. The main contribution of the project is the proof of concept flight test demonstration of a new integrated guidance and control algorithm (see Chapter III). The success of this endeavor had a synergistic effect on several other projects, paving the way for joint ventures in voice controlled flight, coastal mapping, and autonomous landing. The project is viewed as the foundation of a long-term investment in innovative unmanned air vehicle applications leveraging, in part, the operational experience of the officer students in the avionics curriculum.

Testing of a new algorithm, sensor package, vehicle, etc., requires expertise from many branches of the engineering sciences, especially aeronautic, electrical, and computer. It is potentially costly and time consuming, as well as having the potential for catastrophic failure. When successfully done, however, it provides developmental information, insight and data that are unavailable from other sources. All of our theoretical and numerical results must be verified by some form of experiment, and flight testing is often the best way to do this.

The chapter begins with a conceptual discussion of the Rapid Flight Test Prototyping System (RFTPS). Motivation for its development is addressed. Next, a description of the hardware components is given. The main contribution of the chapter is discussed in the last section, where an application of the RFTPS to the problem

of integrated guidance and control introduced in Chapter III is presented. The full capabilities of the RFTPS are demonstrated when this novel guidance algorithm is taken from theoretical development to flight test.

B. SYSTEM DESCRIPTION

The RFTPS consists of a test bed unmanned air vehicle equipped with a complete avionics suite necessary for an autonomous flight as shown in Figure 38, and of a ground station responsible for flight control of the UAV and flight data collection as shown in Figure 39. A functional block diagram of the RFTPS is shown in Figure 40. The key goal was to use off-the shelf-technology as much as possible, thus exploiting the economy of scale of a number of commercial industries. Furthermore, if the UAV development program is to span many years, and to draw on the talents of the officer students in the future, the RFTPS had to emphasize high level algorithm design. Low level code and device driver generation is kept to a minimum with the vast majority of the code "writing" being done via autocode tools. The system architecture is open, providing the ability to add, remove or change real time input/output (I/O). Computational power can be increased as mission requirements dictate. The telemetry links are secure, yet low power and unobtrusive to the public, not requiring advance permission for use, special frequencies from a government authority, or special airspace. The onboard components are light weight and low power, allowing for the inclusion of additional payload.

1. RFTPS Capabilities

The RFTPS developed provides the following capabilities.

- Within the RFTPS environment, one can synthesize, analyze and simulate guidance, navigation, control, and mission management algorithms using a high level development language. The same code that ran the simulation, flies the vehicle.
- Algorithms are seamlessly moved from the high level design and simulation environment to the real time processor.

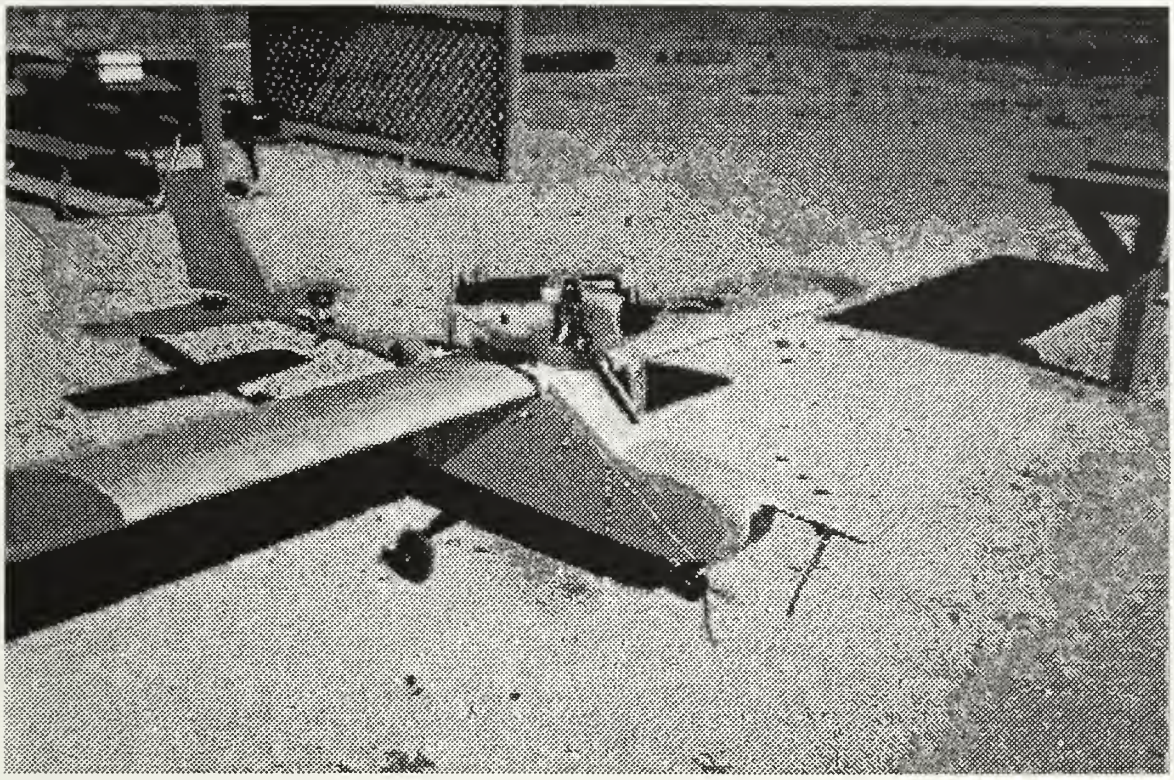


Figure 38. *Frog*: The unmanned air vehicle at the Naval Postgraduate School.

- The RFTPS utilizes industry standard I/O including digital to analog, analog to digital, serial, and pulse width modulation capabilities.
- The RFTPS is portable, easily fitting in a car. In general, testing will occur at fields away from the immediate vicinity of the Naval Postgraduate School.
- The unmanned air vehicle can be flown manually, autonomously, or using a combination of the two. For instance, automatic control of the lateral axis can be tested while the elevator and throttle are controlled manually.
- All I/O and internal algorithm variables can be monitored, collected and analyzed within the RFTPS environment.

2. Cost, Safety and Other Considerations

Cost and risk are two leading, and at times competing, concerns that had to be effectively handled. Since initial testing is to occur within line of sight at all times, a pulse width modulated (PWM) remote control system manufactured by Futaba was chosen. Testing of a new control algorithm is similar to handing over control of the

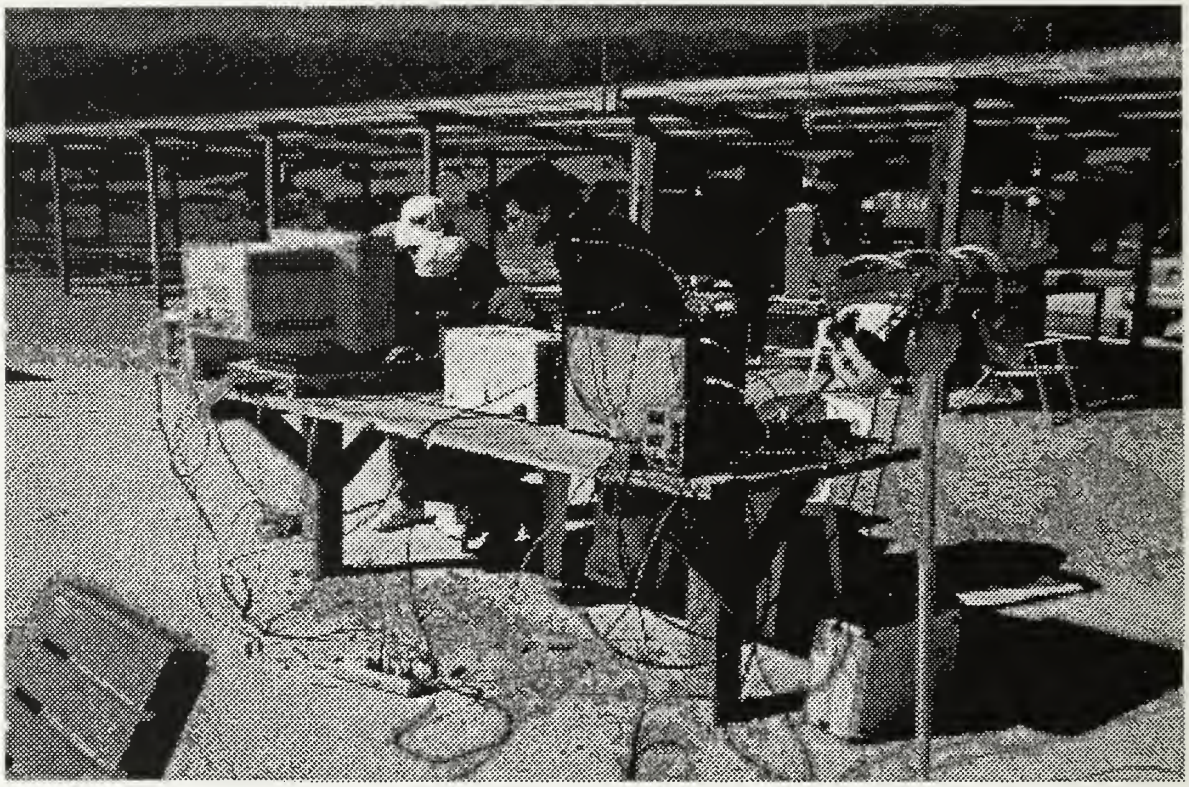


Figure 39. The base station of the RFTPS in use at the airfield.

aircraft to a student pilot. The algorithm should have full freedom to perform, yet adequate safeguards must exist in case it fails. With some modifications, the extensive master-slave flight training capabilities built in to the existing RC transmitters were exploited. A significant portion of the cost of the RFTPS resides in the real time processor, I/O board and modules, and in the host computer. Additionally, while compact, the weight and power requirements of these components are significant when compared to onboard power and payload available. In order to gain additional payload, and in order to manage the risk associated with the loss of an expensive computer package, the real time controller was kept on the ground. Sensor and control links to the real time controller were bridged via RF components described later.

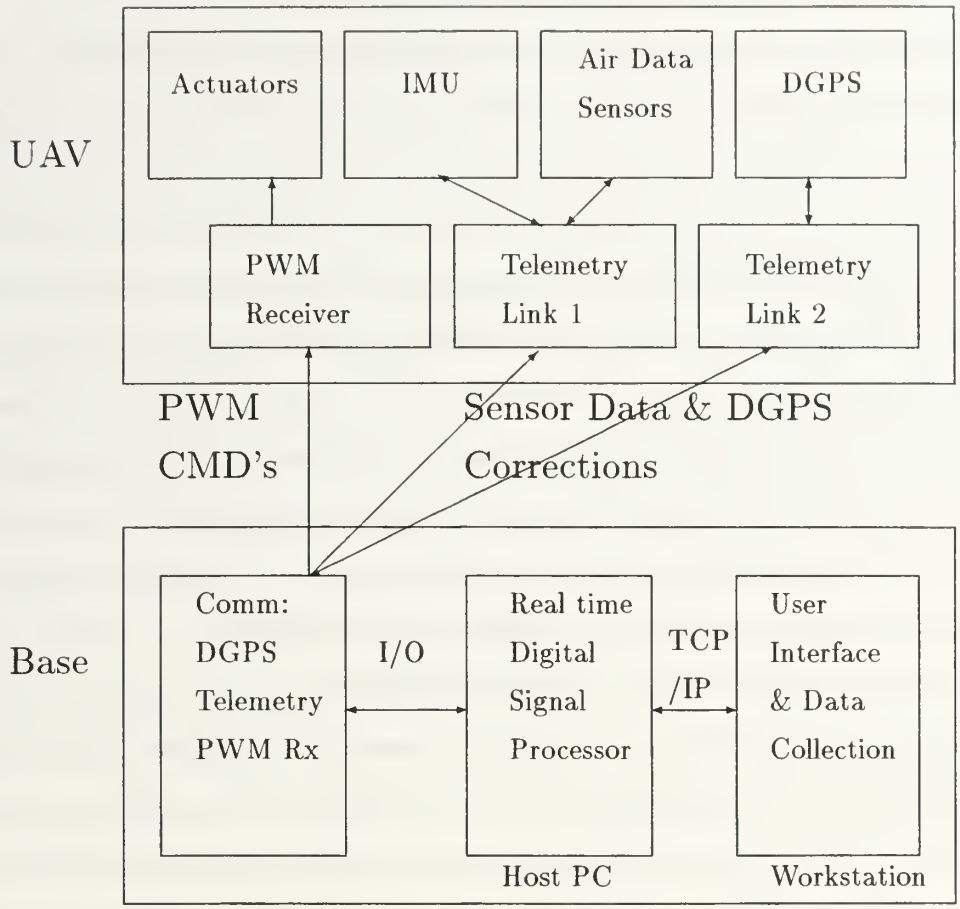


Figure 40. RFTPS Hardware Architecture

3. Components

The centerpiece of the RFTPS ground station is the AC100/C30 system from Integrated Systems Incorporated. The key feature of this product is its autocode tools. With a relatively short time available for research by the officer students, emphasis had to be shifted from code writing, debugging and maintenance to algorithm development. AC100/C30 utilizes "Xmath/SystemBuild", a graphical programming environment that uses a high level block diagram paradigm for modeling of linear and nonlinear systems. Within the "Xmath/SystemBuild" environment, the algorithm can be built, simulated, tested, and debugged. Real-time code can then be

generated for execution on the real time processor.

Currently, the "Xmath/SystemBuild" environment resides on a Sun workstation. PC's running Windows NT are also supported. This allows for a significant downsizing of the ground station should resources become available in the future. Communication with the real time processor is via an Ethernet bus using TCP/IP protocol. AC100/C30 provides excellent animation tools for building graphical user interfaces (GUI). Through the appropriate design of these interfaces, the flight test team can monitor, modify ,and control the actions of the real time processor. The GUI resides on the workstation. Communication between the workstation and the real time processor is via the Ethernet connection, and is managed by a host PC. Additionally, the host PC provides power to the real time processor, as well as providing utilities for compiling, linking, and downloading the C-code.

The I/O consists of four multi-mode, bi-directional, serial ports utilizing RS-232 protocol, a 16 channel pulse width modulation port capable of measuring up to sixteen PWM signals or generating up to six PWM signals, and a six channel digital-to-analog converter. The I/O modules are hosted by the same PC that holds the real time processor. The real time processor is a single Texas Instruments Digital Signal Processor (TMS320C30). The capability exists for upgrading the processor, or running multiple processors, to meet computational demands of future projects.

The control configuration of the air vehicle is conventional with three independent surfaces (elevator, aileron, rudder) and a throttle. Manual control is provided via a Futaba, dual conversion, PWM transmitter utilizing the portion of the radio spectrum reserved for Radio Controlled (RC) flight, 72.030 MHz to 72.990 MHz. Precautions entail a search of the electronic spectrum utilizing a hand held spectrum analyzer, as well as standard procedures employed by RC hobbyists to avoid two individuals selecting the same frequency locally. Built in capabilities of the transmitter include the ability to transmit one or more signals from a slave transmitter. The slave transmitter is a modified Futaba transmitter where the manual control effectors

have been replaced by a direct connection to the digital-to-analog I/O module. In this way, an exogenous source (RFTPS) can be given control of one, some, or all of the control actuators of the aircraft using the same RC link currently controlling the aircraft.

The sensor suite onboard the air vehicle consists of an inertial measurement unit (IMU) with a three axis rate gyro, three axis accelerometer, magnetic heading indicator, two axis pendulum, phase differential GPS receiver, elevator, aileron, and rudder actuator position sensors, and angle of attack, side-slip angle, pitot-static, and static pressure air data sensors. A four channel analog-to-digital converter is used to capture any four of the following sensors: elevator, aileron, rudder actuator position, angle of attack, side-slip angle, dynamic pressure, or static pressure.

Communication between the sensors and the real time processor is via a serial link. Low power, matched, spread spectrum RF links provide up to 115 Kbaud rates at over 10 miles range. They require no license, and can be used anywhere in the United States. Additionally, the onboard GPS unit maintains contact via a serial link with a GPS receiver on the ground, which provides differential corrections.

Power for the onboard avionics is supplied from a lithium-ion battery. The battery provides 6 hours of continuous use. The power budget of the onboard components is shown in Table III.

	Voltage	Current	Power
IMU	24 Volts	400 Milliamps	9.6 Milliwatts
GPS	12 Volts	250 Milliamps	3.0 Milliwatts
Telemetry 1	24 Volts	300 Milliamps	7.2 Milliwatts
Telemetry 2	24 Volts	300 Milliamps	7.2 Milliwatts
Total Req.			27 Milliwatts
Battery			$\geq$ 200 mW-Hrs

Table III. Power Budget of Onboard Components

C. TRAJECTORY CONTROL: AN APPLICATION

The intent of this application was to demonstrate the utility of the RFTPS by flight testing a new integrated guidance and control algorithm that was shown to have good performance and robustness properties both theoretically and in simulation (see II). This project was chosen because, on one hand, it is a totally unique application in terms of the guidance and control algorithms implemented. On the other hand, to implement this algorithm, most of the steps required of any application involving autonomous flight of an air vehicle must be accomplished. If done correctly, the foundation will be laid for follow on projects and joint ventures utilizing unmanned air vehicles.

Generic tasks fundamental to guidance, navigation and control algorithm development were accomplished first. To begin with, a high fidelity model of the test bed vehicle, nicknamed *Frog*, was developed. This involved a complete lateral and longitudinal parameter identification of *Frog's* stability and control derivatives. This lead to the development of a six degree of freedom, nonlinear simulation. In order to manage the risk associated with the autonomous flight, it was decided to add an onboard inner-loop controller. The inner-loop controller modifies the vehicle dynamics such that displacements of the primary longitudinal and lateral control effectors correspond to various *trimming trajectories*, as defined in Chapter III. The inner-loop controller selected was a commercial autopilot whose control laws had to be identified as well. Additionally, accurate and timely calibration of the I/O needed to become an integral part of the RFTPS to account for changing environmental conditions, battery levels, etc.

This section begins with a brief background of the parameter estimation algorithm used to identify a model of *Frog*. Next, the experimental testing completed to obtain data used in the parameter identification process is summarized. Flight test data is compared with the simulation results to demonstrate the efficacy of the effort. Then, a linear controller design based on modeling results obtained is discussed.

Next, a nonlinear controller implemention based on the theory developed in Chapter III is presented. Finally, flight tests conducted using the RFTPS to track an interesting representative trajectory are discussed. Data from flight tests are presented and contrasted with results from simulations. Along the way, implementation issues germane to the flight testing process are explained.

1. Theoretical Background for Model Identification

Parameter identification of the test bed vehicle and inner-loop autopilot was accomplished based on a series of test flights and lab experiments. The Maximum Likelihood Method was chosen to identify the parameters of the model. The background is extracted primarily from [Ref. 18]. Assume that the model to be identified has the following form

$$\begin{aligned} \dot{x} &= A(p)x + B(p)u, \\ y &= C(p)x + D(p)u, \\ x &= \text{state vector}, \\ u &= \text{input vector}, \\ y &= \text{otput vector}, \\ p &= \text{parameter vector}. \end{aligned}$$

The problem at hand is to estimate the parameter vector, p, given a data set of measured inputs, u, and outputs, y. To that end, denote the estimated output based on the parameter vector p as $\hat{y}(p)$, and use it to define a cost function

$$J(p) = \|\hat{y}(p) - y\|_2^2.$$

The method seeks to minimize the cost, J, by varying the parameter vector p. Given an initial guess of the parameter vector p_0, the Jacobean of the cost function J is calculated by numerically perturbing the parameter vector p_0

$$\begin{aligned} \frac{\partial J}{\partial p} &= \frac{\hat{y}(p_0 + \delta p) - \hat{y}(p_0)}{\delta p}, \\ &=: H_0. \end{aligned}$$

Then, a linear approximation for $\hat{y}(p+\delta p)$ is

$$\hat{y}(p+\delta p) = \hat{y}(p_0) + H_0 \delta p.$$

The cost J can now be expressed as a function of the perturbation of the parameter vector as

$$\begin{aligned} J(\delta p) &= \|\hat{y}(p+\delta p) - y\|_2^2, \\ &= \|\hat{y}(p_0) + H_0\delta p - y\|_2^2, \\ &= \|\hat{y}(p_0) - y\|_2^2 + 2\hat{y}(p_0)^T H_0 \delta p - 2\delta p^T H_0^T y + \delta p^T H_0 \delta p. \end{aligned} \tag{IV.1}$$

In order to determine a δp which minimizes equation IV.1, set

$$\begin{aligned} \frac{dJ}{d(\delta p)} &= 2H_0^T \hat{y}(p_0) - 2H_0^T y + 2H_0^T H_0 \delta p, \\ &= 0, \end{aligned}$$

from which it is evident that

$$H_0^T H_0 \delta p = H_0^T (y - \hat{y}(p_0)) \tag{IV.2}$$

must hold.

A common problem encountered in solving a parameter identification problem is the presence of redundant parameters in the model. This results in the matrix $H_0^T H_0$ being singular, and presents a problem if a solution to equation IV.2 is sought in terms of δp. In order to avoid this singularity, the following problem is solved instead,

$$(H_0^T H_0 + \mu I)\delta p = H_0^T (y - \hat{y}(p_0)). \tag{IV.3}$$

With the descent direction δp known, a line search is used to calculate the step size. This method is guaranteed to converge to a minimum for the cost function J, although not necessarily a global minimum [Ref. 18].

The appealing feature of this method for the problem at hand is that it can exploit a great deal information known about the model. For instance, the order of the

plant and most of the structure in the system matrix is known, as is a reasonable range of values for the parameter vector p based on empirical methods (USAF DATCOM).

2. Vehicle Model Identification

The conventional configuration of *Frog* suggested that the parameter identification problem could be decoupled into longitudinal and lateral dynamics. Maximum likelihood parameter identification was used to refine existing analytic estimates of the longitudinal stability and control derivatives [Ref. 34]. The results for a few key longitudinal stability derivatives are compared to analytic estimates in Table IV. The primary difference was in the estimate of the elevator effectiveness, which was less than half as effective has previously thought.

Derivative	Analytic	Experimental	%Δ
C_{L_α}	4.30	4.09	5.13
C_{M_α}	-.417	-.557	-25.1
$C_{L_{\delta e}}$	-1.12	-.391	186
$C_{M_{\delta e}}$	-1.62	-1.05	54.3

Table IV. Comparison of selected longitudinal derivatives.

Figure 41 shows the response, in simulation, of a longitudinal model of *Frog* using these derivatives to an elevator doublet. Comparisons are made to the results from flight test.

Next, the process was repeated for the lateral axis. Aileron and rudder doublets were executed while aileron and rudder position, roll and yaw rates, and side-slip angle were measured. Results, for a few key lateral stability derivatives, are compared to analytic estimates in Table V. The biggest difference was the value of C_{N_β}, which increased by over 300 percent. This doubled the estimate of the natural frequency of the dutch roll mode. Figure 42 shows the response, in simulation, of a lateral model of *Frog* using these derivatives to an aileron doublet. Comparison to the results from flight test are also shown.

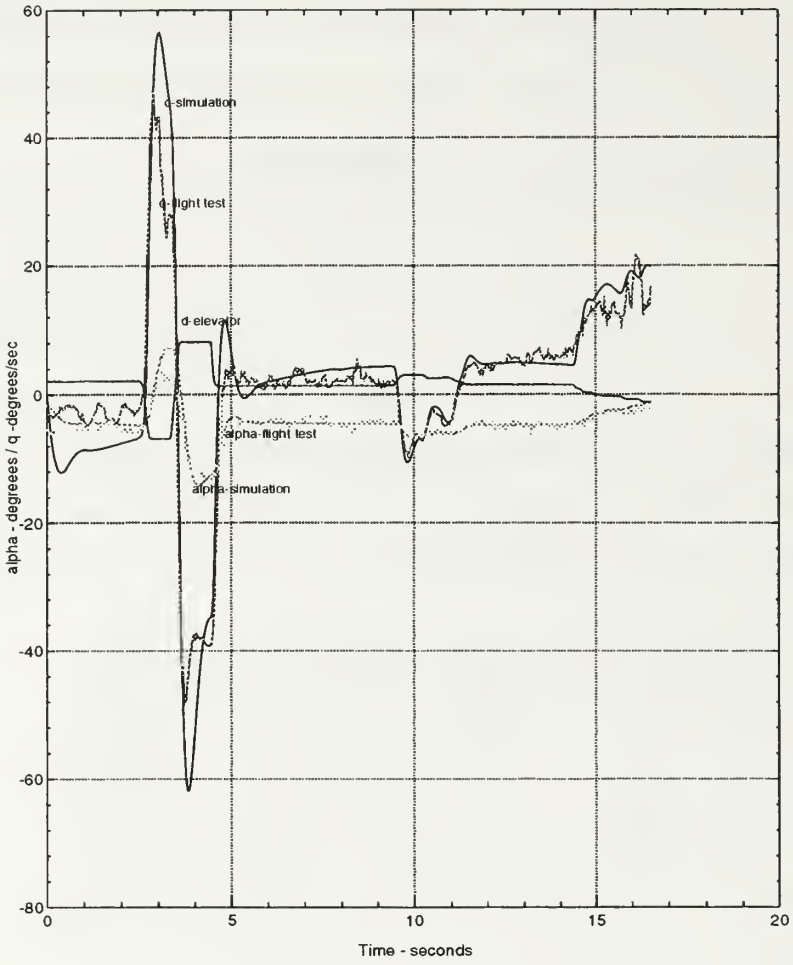


Figure 41. An elevator doublet is used to excite the longitudinal dynamics of *Frog*. Test flight data is compared with simulation results to assess validity of the model.

3. Autopilot Model Identification

The inner-loop autopilot is a "black box" containing both sensors and controller logic. The intent was to model the unit as closely as possible without disassembling it. The published function of the autopilot is to control vertical speed and turn rate of the aircraft. This naturally decouples into an identification problem for the lateral channel, and an identification problem for the longitudinal channel.

The lateral channel employs a rate gyro to track turn rate commands via feedback to the aileron. The general structure is shown in Figure 43. In order to identify the dynamics of the block labeled $\mathcal{T}_1$ in Figure 43, the unit was rotated at

Derivative	Analytic	Experimental	%Δ
C_{Y_β}	-.310	-.987	-68.6
C_{l_β}	-.051	-.094	-45.7
C_{N_β}	.058	.176	-67.0
$C_{l_{\delta a}}$	.181	.239	-24.3

Table V. Comparison of selected lateral derivatives.

differing yaw rates. This provided a variable input signal to the feedback path with a frequency content covering 0 to 20 radians per second. The commanded yaw rate, r_c, was held constant. The feedback loop was broken at the summing junction of r_c and r_{fb}, and the feedback signal was captured. Parameter identification algorithms were used to determine that the feedback loop could be approximated by the following transfer function:

$$\mathcal{T}_1 = \frac{-.1s + 1}{s + 1}. \tag{IV.4}$$

With the autopilot on, a step command in turn rate was transmitted to the vehicle in flight. Vehicle turn rate, as measured by the IMU, was recorded and used to estimate the value of K_{lat} at 0.25. Test data from that flight is shown in Figure 44 and compared with simulation results using the autopilot model.

The longitudinal channel of the autopilot senses the rate of change of static pressure in order to control the vehicle's vertical velocity via feedback to the elevator. Flight tests data capturing the response of the vehicle to a step input in climb rate command was used for model identification. The identified model of the longitudinal channel of the autopilot is shown in Figure 45.

With accurate models of *Frog* and of the onboard autopilot identified, it was a simple matter to determine the command bandwidths of the inner-loop controller (see Figure 46). For this project, this would be the limiting factor in achievable performance of the guidance algorithm. Hardware-in-the-loop testing of the actuators found their bandwidth to be an order of magnitude higher than the inner-loop

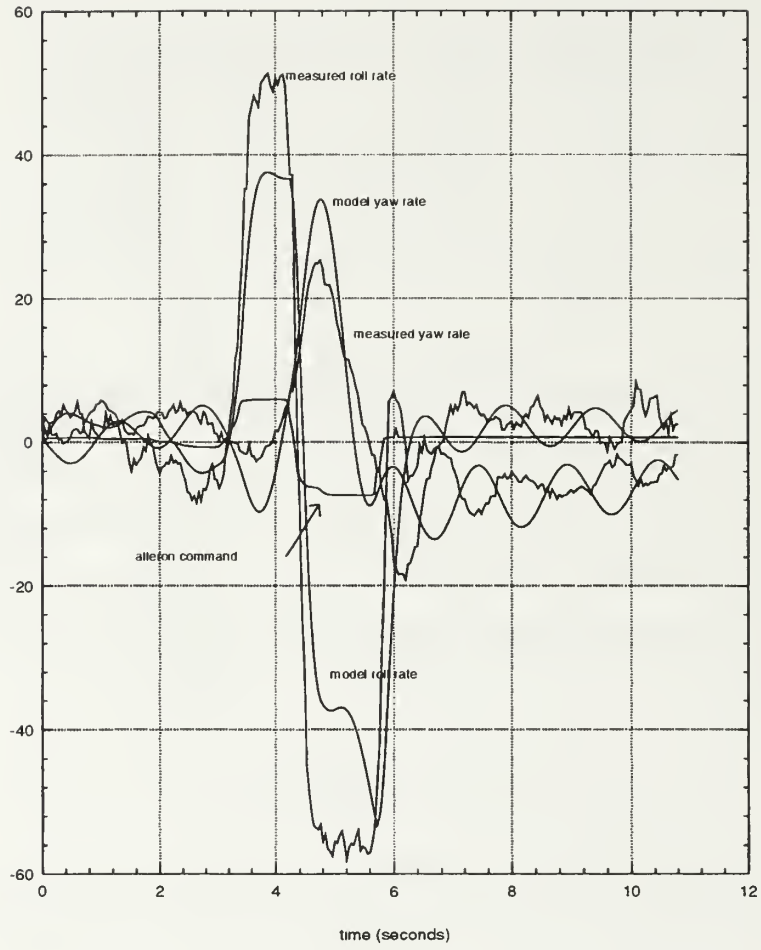


Figure 42. An aileron doublet is used to excite the lateral dynamics of *Frog*. Test flight data is compared with simulation results to asses validity of the model.

command bandwidths. A natural next step would be to remove the autopilot after sufficient time and experience have reduced the risk exposure to acceptable levels, and exploit the extra actuator bandwidth available.

4. Design of the Controller

The design requirements to be met were as follows:

1. **Tracking Requirements**

   - The controller should achieve perfect steady state tracking of trajectories in $\mathcal{E}$ in the presence of a constant disturbance. For a definition of the set $\mathcal{E}$, see Chapter III.

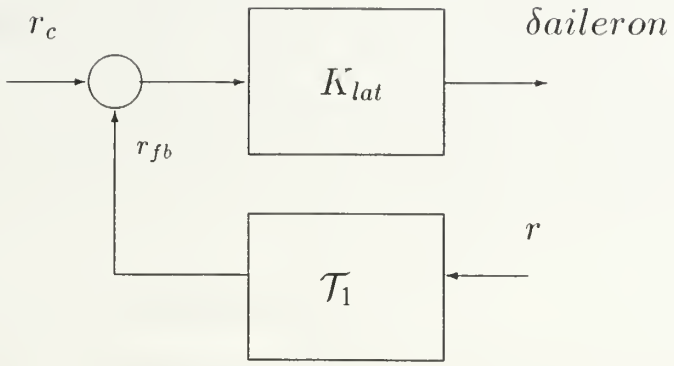


Figure 43. Block diagram of the lateral channel of the inner-loop autopilot.

2. **Bandwidth Requirements**

   - Command bandwidths along the lateral and longitudinal channel should be maximized to ensure tight tracking of the trajectory. Adequate frequency separation between the inner and outer loops should be assured.

3. **Closed Loop Damping and Stability Margins**

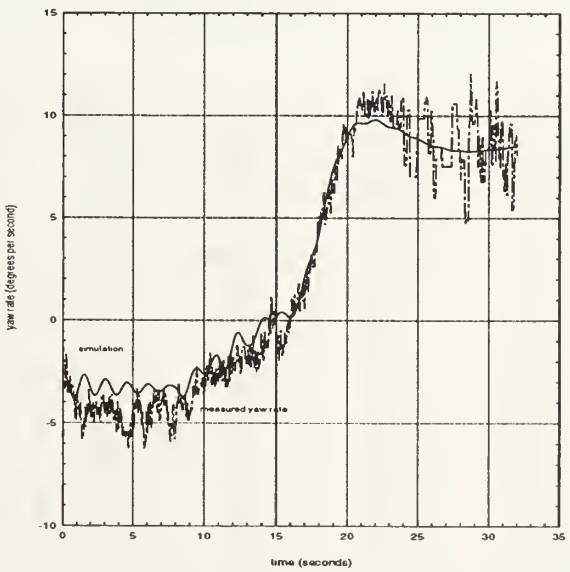


Figure 44. Flight test data is used to identify the dynamics of the autopilot. A step signal is sent to the lateral channel of the autopilot. Measured yaw rate is compared to simulation results.

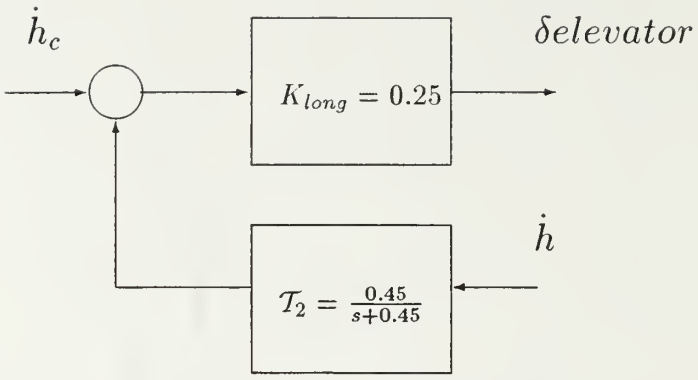


Figure 45. Block diagram of the longitudinal channel of the inner-loop autopilot.

- The dominant closed-loop eigenvalues should have a damping ration of at least 0.5. Phase and gain margins should be no less than 45 degrees and 6 dB respectively in each control loop.

4. **Implementation Requirements**

- Control of *Frog* from the console, open-loop, must be possible in order to maneuver the vehicle to a suitable location overhead the field based on visual cues and displayed navigational data.

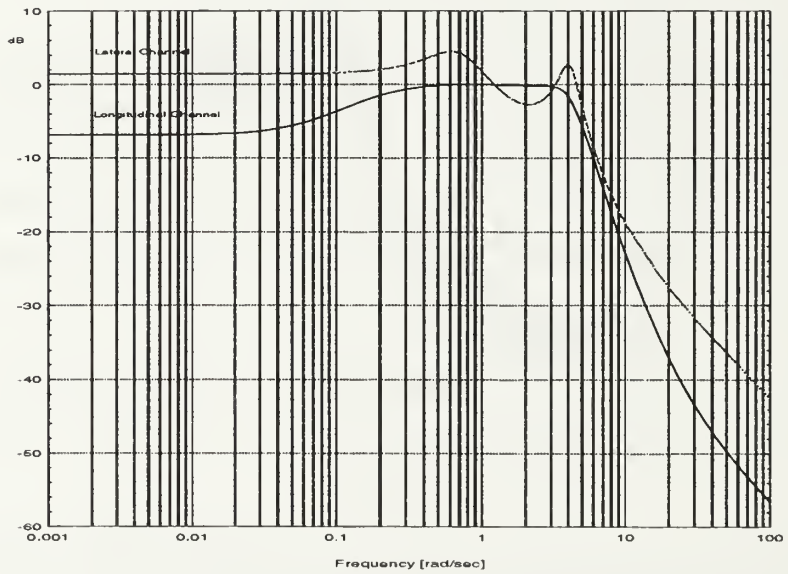


Figure 46. Bode plot of the yaw rate command to yaw rate and climb rate command to climb rate for *Frog* with the autopilot on.

- Switching from open-loop control to autonomous flight should be accomplished from the console at the discretion of the console operator. The switch should be *bumpless* with no undesirable transients.
- The definition of the inertial trajectory passed to the controller should be easily defined from the console, and expressed in terms of the parameters introduced in Chapter III, namely, helix angle (γ_c), radius (b_c), and turn rate ($\dot{\psi}_c$).

a. Linear Controller Synthesis

The synthesis of the controller was an application of the theory outlined in Chapter III where the trimming trajectories were parameterized by the vector $\eta_c = \left[v_c \; \dot{\psi}_c \; \gamma_c\right]^T \in \mathcal{R}^3$. This was useful for the derivation of the generalized error dynamics. For the application at hand, however, it was convenient to remove the explicit dependence on v_c. Since along $P_C \in \mathcal{E}$

$$b_c = \frac{v_c \cos(\gamma_c)}{\dot{\psi}_c},$$

there is no problem in using $\bar{\eta}_c = \left[b_c \; \dot{\psi}_c \; \gamma_c\right]^T \in \mathcal{R}^3$ to parameterize $P_C \in \mathcal{E}$ vice η_c. This was done strictly for convenience, since during the flight test, the throttle was to be left at a cruise power setting, and the airspeed was not explicitly controlled. Flight tests tracked a single trajectory in $\mathcal{E}$ identified as

$$\bar{\eta}_c = \left[\begin{array}{ccc} 1146 & 5 & 0 \end{array}\right]^T, \tag{IV.5}$$

where the units are feet, degrees and seconds, respectively. Based on (III.6), (III.7) and the model identification results in section 2 and 3, the nonlinear plant,

$$\mathcal{G} = \left\{ \begin{array}{ll} \frac{d}{dt} V & = \mathcal{F}_V(V,\Omega,\Lambda) + \mathcal{I}_V(V,\Omega)\mathcal{H}(V,\Omega,U), \\ \frac{d}{dt} \Omega & = \mathcal{F}_\Omega(V,\Omega,\Lambda) + \mathcal{I}_\Omega(V,\Omega)\mathcal{H}(V,\Omega,U), \\ \frac{d}{dt} P & =^I_B RV, \\ \frac{d}{dt} \Lambda & = \mathcal{Q}\,\Omega, \end{array} \right. \tag{IV.6}$$

was available for controller synthesis. Given (IV.5) and (IV.6), the trimming values, V_C, Ω_C, and U_C were solved as the solution to

$$\mathcal{F}_V(V_C,\Omega_C,\Lambda_C) + \mathcal{I}_V(V_C,\Omega_C)\mathcal{H}(V_C,\Omega_C,U_c) = 0,$$

$$
\begin{aligned}
&\mathcal{F}_\Omega(V_C, \Omega_C, \Lambda_C) \; + \; \mathcal{I}_\Omega(V_C, \Omega_C)\mathcal{H}(V_C, \Omega_C, U_c) = 0, \\
&\mathcal{Q}^{-1}\Omega_C - \dot{\Lambda}_C = 0, \\
&\|V_C\| - \frac{b_c\dot{\psi}_c}{\cos(\gamma_c)} = 0 \\
&\beta_c - \sin^{-1}\frac{\frac{b_c\dot{\psi}_c}{\cos(\gamma_c)}}{V_t} = 0, \\
&\gamma_c - [0 \;\; 1 \;\; 0]\, arg({}^W_B\mathcal{R}^I_B R) = 0,
\end{aligned}
\tag{IV.7}
$$

where the arg function extracts the angles X from the rotation matrix $\mathcal{R}(X)$: $X = arg(\mathcal{R}(X))$. Numerically solving (IV.7), the trim values were found to be

$$
\begin{aligned}
V_C^T &= \begin{bmatrix} 87.7 & 2.037 & 2.43 \end{bmatrix}^T, \\
\Omega_C^T &= \begin{bmatrix} 0.00176 & 0.029 & 0.087 \end{bmatrix}^T, \\
\Lambda_C^T &= \begin{bmatrix} 13.41 & 0.023 & 0 \end{bmatrix}^T,
\end{aligned}
$$

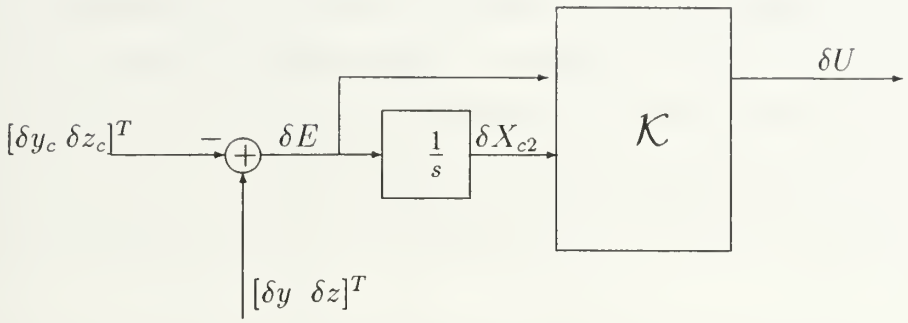
where, as before, standard units are feet, degrees and seconds.

Based on lab and flight testing of the onboard sensors, the decision was made to design an output feedback controller of the form (see Figure IV.8):

$$
\mathcal{C}_l = \begin{cases}
\delta E &= [\delta y \;\; \delta z]^T - [\delta y_c \;\; \delta z_c]^T, \\
\frac{d}{dt}\delta X_{c1} &= \mathcal{A}_{c1}\delta X_{c1} + \mathcal{B}_{c2}\delta X_{c2} + \mathcal{B}_{c3}\delta E, \\
\frac{d}{dt}\delta X_{c2} &= \delta E, \\
\delta U &= \mathcal{C}_{c1}\delta X_{c1} + \mathcal{D}_{c2}\delta X_{c2} + \mathcal{D}_{c3}\delta E,
\end{cases}
\tag{IV.8}
$$

where $\delta U = \begin{bmatrix} r_c & \dot{h}_c \end{bmatrix}^T$.

Classical control techniques were employed to design the controller, $\mathcal{C}_l$, shown in Figure 47. Performance objectives included setting the loop gain crossover frequencies at 0.5 radians per second along each channel based on the inner-loop command bandwidths (see Figure 46). Root-locus analysis along the lateral channel provides the most intuitive view of the control problem (see Figure 48). The open-loop plant has unstable zeros along the lateral channel due to the dihedral effect of



$$\mathcal{K} = \left[\begin{array}{c|cc} A_{c1} & B_{c2} & B_{c3} \\ \hline C_{c1} & D_{c2} & D_{c3} \end{array}\right]$$

Figure 47. Linear Controller Design

the high wing design. These unstable zeros will naturally attract the free integrators in the synthesis model if left unattended.

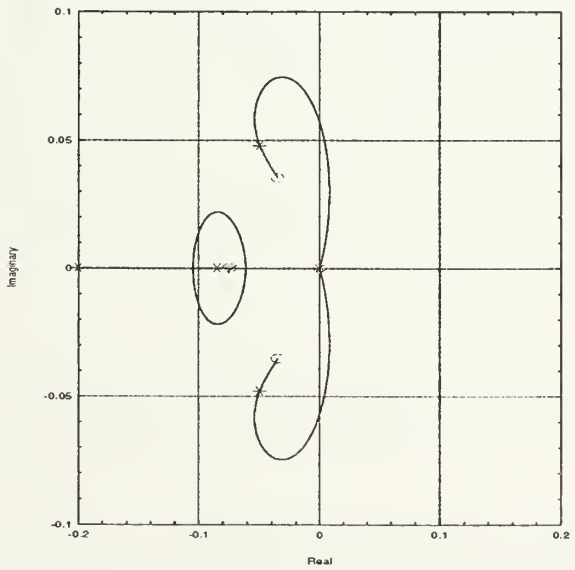


Figure 48. Root-locus for the lateral channel.

The straightforward solution was to interlace the pole-zero structure of the synthesis model with a stable pair of complex zeros sufficiently close to the

free integrators. The robustness limit was set by the dutch roll poles which were forced to eventually migrate to the non-minimum phase zeros. Modification of the dutch roll behavior would require a different approach than that chosen with the inclusion of the inner-loop autopilot. Replacement of the inner-loop autopilot is a natural extension for future flight test work. Design concerns and solutions along the longitudinal channel were qualitatively similar. The resulting design was a sixth order controller.

Nyquist diagrams of the loop transfer function for the lateral and longitudinal channels are shown in Figure 49 and 50. There it can be seen that phase and gain margin design requirements along each channel have been met.

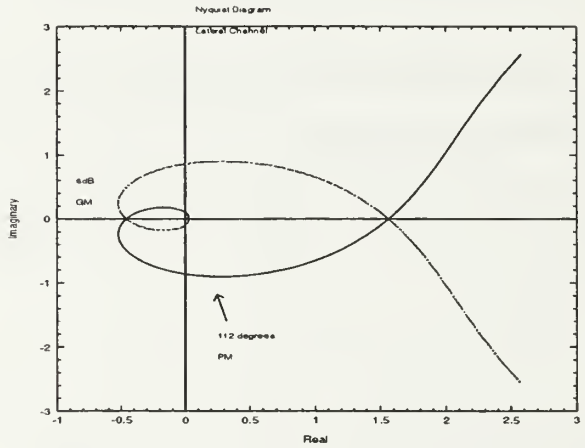


Figure 49. Nyquist diagram of the loop transfer function for the lateral channel. The phase margin is 112 degrees and the gain margin is 6 dB.

5. Nonlinear Implementation

The controller designed in the preceding section was implemented on the nonlinear plant using the implementation proposed in Chapter III and is given next (see

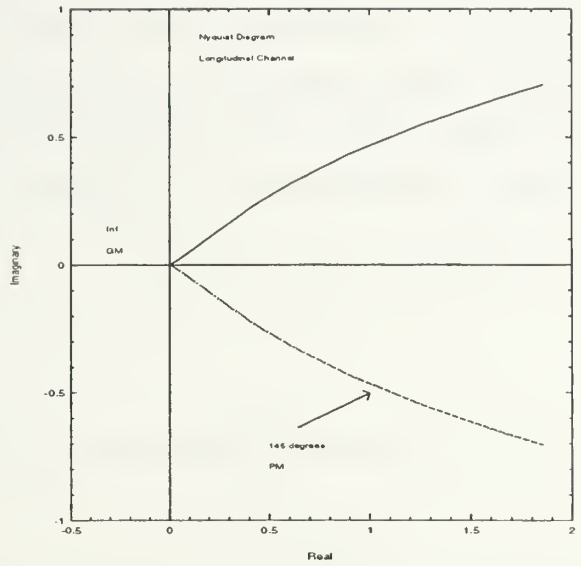


Figure 50. Nyquist diagram of the loop transfer function for the longitudinal channel. The phase margin is 145 degrees and the gain margin is infinite.

Figure 51):

$$\mathcal{C} := \begin{cases} [0 \;\; y \;\; z]^T &= {}^{A}_{I}\mathcal{R}(P - P_C(s_0)), \\ E &= [y - y_c \;\; z - z_c], \\ \frac{d}{dt}X_{c1} &= \mathcal{A}_{c1}X_{c1} + \mathcal{B}_{c2}E + \mathcal{B}_{c3}\frac{d}{dt}E, \\ \frac{d}{dt}X_{c2} &= \mathcal{C}_{c1}X_{c1} + \mathcal{D}_{c2}E + \mathcal{D}_{c3}\frac{d}{dt}E, \\ U &= X_{c2}. \end{cases} \tag{IV.9}$$

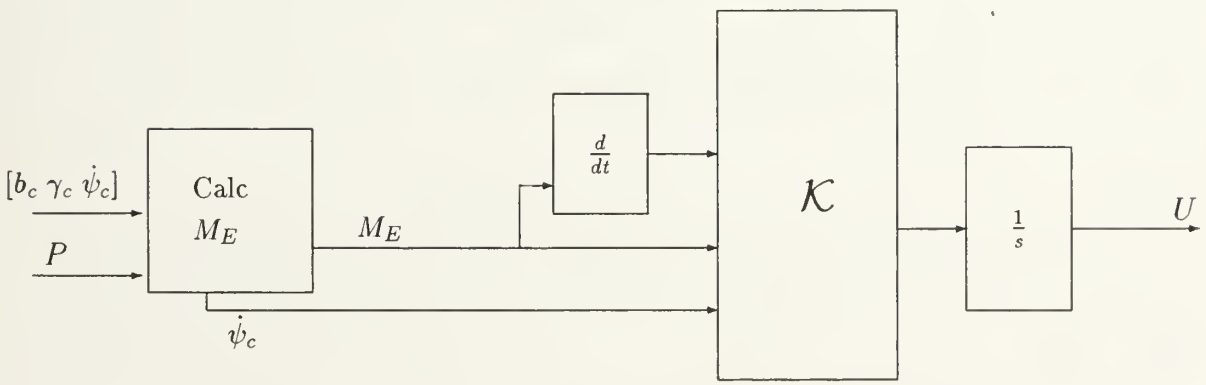


Figure 51. Nonlinear Implementation of the Controller

The following algorithm was implemented in the block labeled, *Calc* M_E, in Figure 51 in order to compute M_E. Assume the position of the vehicle is $[x_l,\ y_l\ z_l]^T$, and let the center of the trajectory be specified as $[x_c,\ y_c\ z_c]^T$. Then, the projection of the position of the vehicle onto the trajectory, as defined by $\bar{\eta}_c$, is

$$P_c(s_0) \ = \ \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^T,$$

where

$$\begin{aligned} x_0 &= b_c \cos(\frac{s_0 \cos(\gamma_c)}{b_c}) + x_c, \\ y_0 &= b_c \sin(\frac{s_0 \cos(\gamma_c)}{b_c}) + y_c, \\ z_0 &= \dot{\psi}_c b_c s_0 \tan(\gamma_c) + z_c. \end{aligned} \tag{IV.10}$$

When γ_c is zero, the parameter, s_0, can be computed by considering the projection of P onto a co-level circle with radius, b_c, and center $[x_c,\ y_c\ z_l]^T$. Then,

$$s_0 \ = \ b_c \tan^{-1}(\frac{y_l - y_c}{x_l - x_c}). \tag{IV.11}$$

The position error resolved in the Frenet frame attached at the point $P_C(s_0)$ is

$$\begin{aligned} \begin{bmatrix} x_{err} \\ y_{err} \\ z_{err} \end{bmatrix} &= \begin{bmatrix} -\cos(\gamma_c)\sin(\frac{s_0}{b_c}) & \cos(\gamma_c)\cos(\frac{s_0}{b_c}) & -\sin(\gamma_c) \\ -\cos(\frac{s_0}{b_c}) & -\sin(\frac{s_0}{b_c}) & 0 \\ \sin(\gamma_c)\sin(\frac{s_0}{b_c}) & -\sin(\gamma_c)\cos(\frac{s_0}{b_c}) & \cos(\gamma_c) \end{bmatrix} \begin{bmatrix} x_l - x_0 \\ y_l - x_0 \\ z_l - z_0 \end{bmatrix}, \\ &=: \ M_E. \end{aligned} \tag{IV.12}$$

If the helix angle is not zero, then a simple solution is not available. The point can be found, however, by noting that M_E must be of the form $[0\ y_{err}\ z_{err}]^T$. Then using (IV.12),

$$-\cos(\gamma_c)\sin(\frac{s_0}{b_c})(x_l - x_0) + \cos(\gamma_c)\cos(\frac{s_0}{b_c})(y_l - y_0) - \sin(\gamma_c)(z_l - z_0) \ = \ 0, \tag{IV.13}$$

must hold. Using (IV.10), (IV.13) is rewritten as

$$\begin{aligned} -\cos(\gamma_c)\sin(\frac{s_0}{b_c})(x_l - b_c\cos(\frac{s_0\cos(\gamma_c)}{b_c}) + x_c) + \\ \cos(\gamma_c)\cos(\frac{s_0}{b_c})(y_l - b_c\sin(\frac{s_0\cos(\gamma_c)}{b_c}) + y_c) - \\ \sin(\gamma_c)(z_l - \dot{\psi}_c b_c s_0\tan(\gamma_c) + z_c) &= 0, \end{aligned} \tag{IV.14}$$

which can be solved for s_0.

D. CONTROLLER IMPLEMENTATION FOR FLIGHT TEST

During a flight test it is critical that the transition from manual to autonomous flight occur without any undesirable transients. The method chosen to achieve this was to initiate autonomous flight with the vehicle on the trajectory, and with the controller states recruited to their nominal values. An elegant approach toward partial completion of this task was to restructure the controller using the $\mathcal{D}$-Implementation. One important convenience of this structure is the straightforward manner in which the controller states can be recruited at the initiation of autonomous flight based on the trajectory definition parameter $\bar{\eta}_c$. Let t_{on} be the time that the console operator turns the controller on. Then, setting

$$\begin{aligned} X_{c2}(t_{on}) &= \begin{bmatrix} \dot{\psi}_c\cos(\gamma_c) \\ \dot{\psi}_c b_c\tan(\gamma_c) \end{bmatrix}, \\ X_{c1}(t_{on}) &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \end{aligned} \tag{IV.15}$$

perfectly recruits the controller states at the initiation of autonomous flight. The structure in (IV.9) is also convenient for implementing hard limits on the feedback signal. It was experimentally determined that the command signals to the autopilot

should be kept within the following ranges

$$\begin{aligned} -15 \text{ degrees per second} &\leq b_c \leq +15 \text{ degrees per second}, \\ -2000 \text{ feet per minute} &\leq \dot{h}_c \leq +2000 \text{ feet per minute}. \end{aligned} \tag{IV.16}$$

Since the feedback signals are the outputs of integrators in (IV.9), the states X_{c2} were easily hard limited to the values in (IV.16), with anti-windup implemented on the associated integrators.

To affect a smooth transition, all that remained was to initiate autonomous flight when the vehicle was on the reference trajectory and aligned with it. Since there were no mission requirements regarding the center of the trajectory, the coordinates were computed to place the vehicle on the trajectory at initiation of autonomous flight. Furthermore, the orientation of the Frenet frame was matched to the vehicle's heading. Let ψ_{on} be the vehicle heading at t_{on}. Let the trajectory parameter be specified by (IV.5). Then, setting

$$\begin{aligned} x_c &= x_{on} - \frac{\cos(\psi_l)\cos(\gamma_c)}{b_c}, \\ y_c &= y_{on} - \frac{\sin(\psi_l)\cos(\gamma_c)}{b_c}, \\ z_c &= z_{on} \end{aligned}$$

defines a trajectory where M_E at t_{on}, as defined by (IV.12), is zero, and $\Lambda_3 = \Lambda_{C_3}$, where the subscript indicates the third element of the vector.

1. Additional Implementation Issues

Prior to autonomous flight of *Frog*, it was decided to test the RFTPS by flying *Frog* remotely from the workstation console. The console operator was provided with the appropriate displays showing *Frog's* position, heading, and velocity, and used that information to command the vehicle's turn rate and climb/descent rate in order to keep *Frog* in the local operating area.

The vehicle's position, as reported by the onboard GPS, is in units of latitude, longitude and height above mean sea level. In order to provide a more intuitive

navigational reference frame, the location of the workstation at the field where the flight tests were conducted was surveyed. This postion was used to define the origin of a local tangent plane. Let $[\lambda_1,\ \lambda_2,\ h]^T$ be the vehicle's position, as reported by the GPS receiver, where

$$\begin{aligned} \lambda_1 &= \text{degrees of latitude,} \\ \lambda_2 &= \text{degrees of longitude,} \\ h &= \text{height GPS in meters.} \end{aligned} \tag{IV.17}$$

The variable h is a height referenced to the surface of an ellipsoid approximating the shape of the earth. Two important parameters describing this reference ellipsoid, termed $WGS84$, are its eccentricity factor (ϵ) and the length of its semi-major axis (a). The distance from the origin of the $WGS84$ ellipsoid to a point on its surface where the local latitude is λ_1 is given by

$$N = \frac{a}{\sqrt{1 - \epsilon^2 \sin^2(\lambda_1)}}. \tag{IV.18}$$

Consider a Cartesian coordinate system with its origin at the center of the WGS84 ellipsoid, oriented such that its z-axis is aligned due north, its x-axis intersects the prime meridian, and its y-axis completes the right hand rule. This reference coordinate system is termed Earth Centered Earth Fixed (ECEF). Then, given the position reported by GPS (IV.17), the position expressed in the ECEF reference frame is

$$\begin{aligned} x &= (N + h)\cos(\lambda_2)\cos(\lambda_1). \\ y &= (N + h)\cos(\lambda_2)\sin(\lambda_1). \\ z &= (N(1 - \epsilon^2) + h)\sin(\lambda_2). \end{aligned}$$

Let $[\lambda_{1_0}\ \lambda_{2_0}\ h]^T$ be the surveyed position of the workstation, and let $[x_0,\ y_0,\ z_0]^T$ be the same location expressed in ECEF coordinates, and suppose this position is used

to define the origin of a local tangent plane reference frame. Sign convention and orientation of the local tangent plane reference frame are defined as positive x-axis values extending due north, positive y-axis values extending due east, and positive z-axis values extending straight down. Furthermore, assume $[x_{frog}\ y_{frog}\ z_{frog}]^T$ is the reported position of the vehicle expressed in the ECEF coordinate system. Then,

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} -\sin(\lambda_{2_0})\cos(\lambda_{1_0}) & -\sin(\lambda_{2_0})\sin(\lambda_{1_0}) & \cos(\lambda_{2_0}) \\ -\sin(\lambda_{1_0}) & \cos(\lambda_{1_0}) & 0 \\ -\cos(\lambda_{2_0})\cos(\lambda_{1_0}) & -\cos(\lambda_{2_0})\sin(\lambda_{1_0}) & -\sin(\lambda_{2_0}) \end{bmatrix} \begin{bmatrix} (x_{frog} - x_0) \\ (y_{frog} - y_0) \\ (z_{frog} - z_0) \end{bmatrix} \tag{IV.19}$$

is the position of the vehicle in the local tangent plane reference frame. This position was displayed on the console, and later, the trajectories tracked by *Frog* were defined in this reference frame.

The final issue that needed to be resolved was the accurate and timely calibration of the command signals from the console to the vehicle. The autopilot onboard the vehicle expected to see commands in terms of PWM signals. The pulse width modulated signals were generated by first converting the command signals to analog voltages via the Digital-to-Analog converter. Then the analog voltages were directed to a Futaba transmitter, specially modified to respond to analog voltages instead of the movement of the exterior joysticks. Next, a receive module onboard *Frog* converted the transmitted signals to PWM format, where they were sent to the autopilot. A functional block diagram of the architecture used to calibrate the uplink is shown in Figure 52.

The key to calibrating the uplink was the determination of the functions F_{pwm}^{-1} (Block 1) and G_{volt}^{-1} (Block 2) in Figure 52. This was done in two steps. The first step involved sending constant commands to the DAC from the user interface (Block 10, 11 and 12). This resulted in flight patterns consisting of numerous constant rate turns to the left and to the right, and to constant rate climbs and descents. A receive module, which was a duplicate of the one onboard *Frog*, converted the

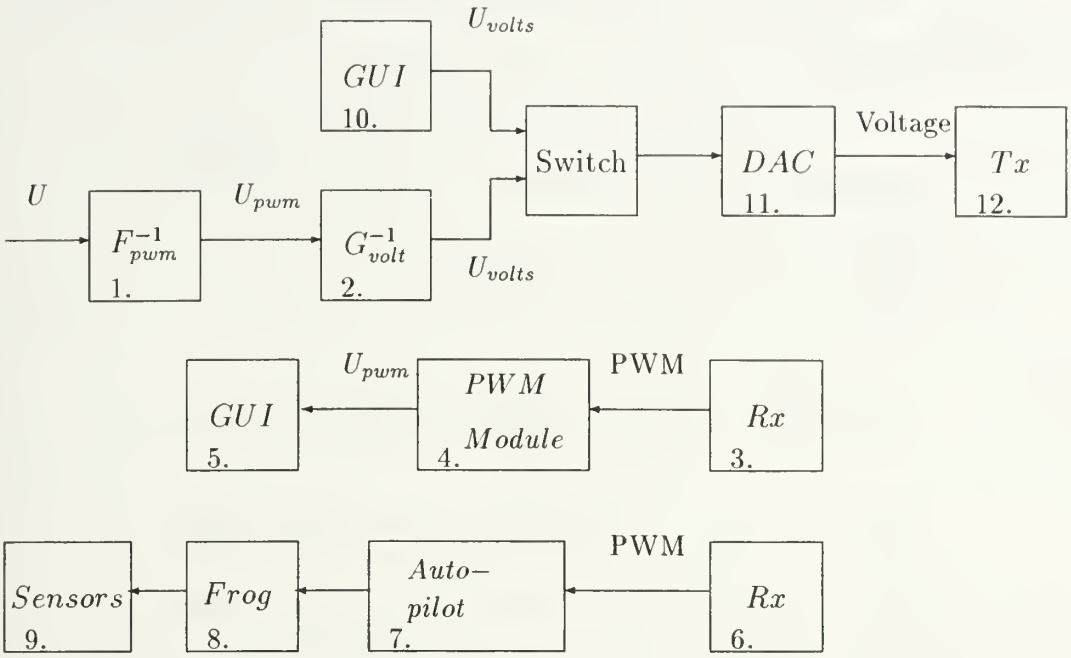


Figure 52. Calibrating the Uplink

transmitted signal to PWM format. This was then converted and stored in digital format by the RFTPS via the pulse width modulated signal converter I/O module (Block 3, 4 and 5). Concurrently, the onboard navigation sensors transmitted yaw rate and climb rate data, among other things, to the RFTPS (Block 6, 7, 8 and 9). Post processing of the data revealed linear relationships between the PWM signals sent to the autopilot and the respective climb/descent or turn rate of the vehicle (see Figure 53). The relationships between the command signals in terms of their PWM values ($r_{c_{pwm}}$ and $\dot{h}_{c_{pwm}}$) and the steady-state performance of *Frog* in flight are expressed as

$$\begin{aligned} r &= 0.0885 r_{c_{pwm}} - 151.476, && \text{(IV.20)} \\ &=: F_{r_{pwm}}(r_{c_{pwm}}), \\ \dot{h} &= 12.3929 \dot{h}_{c_{pwm}} - 18193, && \text{(IV.21)} \\ &=: F_{\dot{h}_{pwm}}(\dot{h}_{c_{pwm}}), && \text{(IV.22)} \end{aligned}$$

where r is the vehicle's yaw rate in degrees per second, and $\dot{h}$ is the vehicle's climb/descent rate in feet per minute.

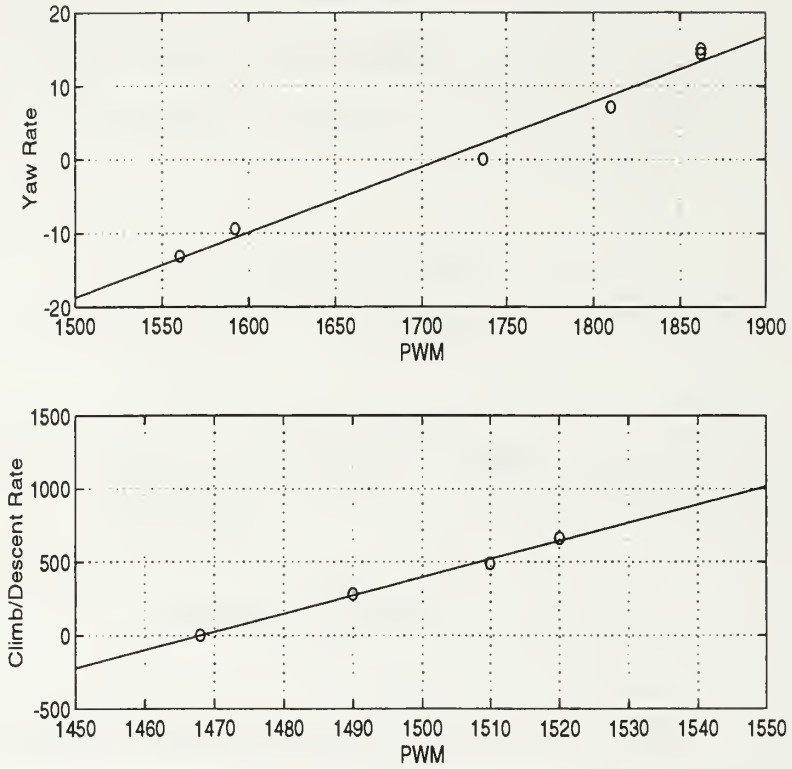


Figure 53. Top graph: measured vehicle yaw rate (degrees per second) versus PWM signal (μ seconds) into the lateral channel of the inner-loop autopilot. Bottom graph: measured vehicle climb rate (feet per minute) versus PWM signal (μseconds) into the longitudinal channel of the inner-loop autopilot.

The second step in calibrating the uplink involved the modified Futaba transmitter (Block 2). The known voltages applied to the transmitter were recorded, while the duplicate receive module was used to decode the transmitted signal, and the RFTPS was used to record the corresponding PWM value (Block 3, 4 and 5). The relationships were linear and expressed as

$$r_{c_{pwm}} = 686.7 r_{volt} - 234.1, \qquad \text{(IV.23)}$$

$$=: G_{r_{volt}}(r_{volt}),$$

$$\dot{h}_{c_{pwm}} = 800 \dot{h}_{volt} - 470, \qquad \text{(IV.24)}$$

$$=: \; G_{\dot{h}_{volt}}(\dot{h}_{volt}),$$

where $\dot{h}_{volt}$ is voltage sent to the longitudinal channel of the modified Futaba transmitter, and r_{volt} is the voltage sent to the lateral channel.

That completed the calibration of the uplink. Given a commanded climb/descent rate ($\dot{h}_c$) in feet per minute, and commanded yaw rate (r_c) in degrees per second at the console, the voltages sent to the modified transmitter were calculated as

$$\dot{h}_{volt} = G^{-1}_{\dot{h}_{volt}} F^{-1}_{\dot{h}_{pwm}} \dot{h}_c, \tag{IV.25}$$

$$r_{volt} = G^{-1}_{r_{volt}} F^{-1}_{r_{pwm}} r_c. \tag{IV.26}$$

E. RESULTS AND ANALYSIS

This section summarizes the results of two flight tests of the guidance and control algorithm developed in the preceding section. While the trajectory definition parameter was identical for both tests, the test flights took place on different days with very different environmental conditions. Furthermore, on the first test, only the lateral channel of the guidance algorithm was active. Longitudinal control was done open-loop by the console operator. Both flights took place at approximately 500 feet of altitude above ground level. Wind measurements were made at ground level on the runway, and later incorporated into the simulation. The pilot maintained control of the throttle throughout the tests. During autonomous flight, the pilot left the throttle at the trim setting for the trajectory defined. The rudder was not used.

At the time of the tests, the RFTPS had been in operation for approximately three months. The two flight tests presented were not the first time the RFTPS was used to control *Frog*. Some of the model identification process, and most of the calibration process, used the RFTPS to control *Frog*. This was primarily due to the fact that the pilot used a conventional stick configuration to control *Frog*. Longitudinal signals corresponded to fore and aft motion, and lateral commands corresponding to left and right motion. Movement along one axis invariably corrupted the signal along the other axis. In contrast, control signals generated by the RFTPS

were very precise and perfectly decoupled. Also, tests had been successfully completed whereby a portable computer was used to send commands to the RFTPS and control *Frog*. The flight tests presented do represent the first time that a guidance and control algorithm was used to control *Frog* fully autonomously.

1. First Flight Test

The first test of the guidance algorithm occurred on June 20, 1997. The wind was measured at 10 mph out of the northwest with gusts to 15 mph. The pilot launched *Frog* and brought it up to a safe altitude. Control was passed to the console operator who turned the controller on. The trajectory definition is repeated for convenience,

$$\begin{aligned} \gamma_c &= 0 \text{ degrees}, \\ \dot{\psi}_c &= 5 \text{ degrees per second}, \\ b_c &= 1146 \text{ feet}. \end{aligned}$$

Autonomous control of the longitudinal channel was not implemented. The GPS operated in the differential position mode throughout the test. Prior testing of this GPS revealed that position error in the horizontal plane, using GPS in differential mode, had a standard deviation of 10 feet. The positions shown are those reported by GPS. The true position is unknown but thought to be within about 10 feet of that shown. Later, the process was repeated in simulation using the measured value of the wind. The trajectory flown by the vehicle is shown projected onto the horizontal plane in Figure 54. Additionally, the reference trajectory and the trajectory obtained in simulation is shown.

The controller activity in the lateral channel is shown alongside the lateral error signal in Figure 55. According to the orientation of the Frenet frame, the error signal went negative when the vehicle was inside of the reference trajectory. The controller activity has noticeable high frequency content caused by the interaction of the one second update rate of the GPS and the slow complex zeros placed in

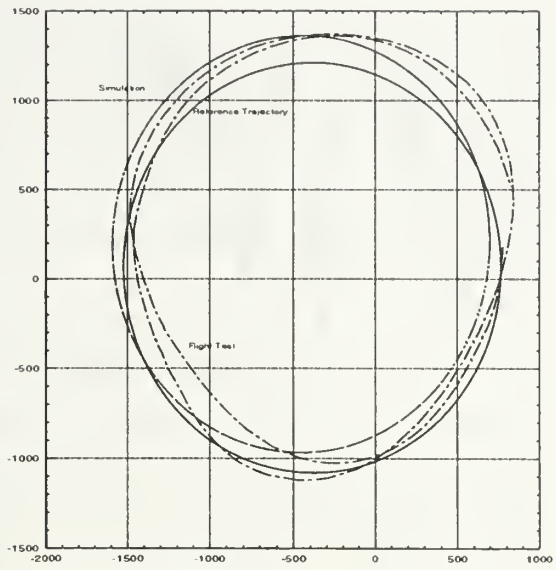


Figure 54. The first (near) autonomous flight of *Frog*. The commanded trajectory is shown as a solid line while the path flown is shown as a broken line. Also shown are the results from nonlinear simulation. The wind was 18 feet per second out of the west, northwest. The graph is oriented with the y-axis pointing due north. Approximately two and one half minutes of flight are shown.

the controller. The nominal commanded yaw rate should be 5 degrees per second, but instead has a mean value of 8 degrees per second. This is indicative of the integral action compensating for modeling and calibration errors. The ground speed (not shown) oscillated by twice the magnitude of the measured wind. The vehicle's indicated airspeed was nearly constant. Therefore, the wind appeared as a sinusoidal disturbance at a frequency of 0.796 radians per second. As the loop gain cross over frequency was 0.5 radians per second along the lateral channel, little gain was available to suppress this disturbance.

Similar qualitative and quantitative performance was observed in simulation. The slight difference in orientation of the two trajectories is most likely due to misalignment of the modeled and actual wind. Table VI quantitatively compares flight test data to linear and nonlinear simulation results. The close agreement in average (RMS) tracking error indicates that the design has good robustness properties. The

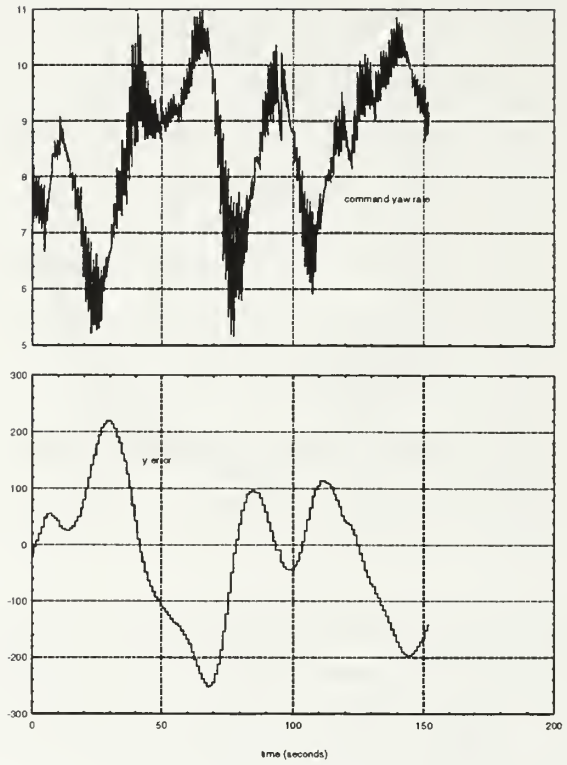


Figure 55. Top graph: Commanded vehicle yaw rate in degrees per second from the controller along the trajectory, flown on June 20, 1997. Bottom graph: Lateral error in feet, as computed by the guidance algorithm, along the same trajectory.

relatively large tracking errors are indicative of the low loop gain available for disturbance rejection of wind effects. In retrospect, performance requirements should be formulated to ensure adequate gain at these frequencies. The frequency of this disturbance, ω_d, is well known at the design stage. It can be defined using $\bar{\eta}_c$ as

$$\omega_d = \frac{360}{\dot{\psi}_c \cos(\gamma_c)}. \tag{IV.27}$$

With estimates of the maximum magnitude of the wind, and requirements on the maximum permissible tracking errors allowed, a performance requirement could naturally be formulated as an H_∞ constraint.

	Linear Simulation	Nonlinear Simulation	**Flight Test**
Lateral Mean Error	-21.92 feet	-22.1 feet	-18.35 feet
Lateral RMS Error	111.27 feet	111.49 feet	121.24 feet
Lateral Peak Error	140.57 feet	140.01 feet	219.54 feet

Table VI. Error Comparison for Flight of June 20, 1997: Flight Test & Simulation

2. Second Flight Test

On July 23, 1997, a second flight test was conducted. The second flight test provided the opportunity to test the guidance algorithm in near ideal weather conditions. The wind was 1 to 2 miles per hour out of the west, and the air was absent of thermal activity and disturbances. This time, the longitudinal channel was operative.

Frog was brought up to altitude by the pilot and control handed over to the console operator. The console operator switched to autonomous flight with the trajectory parameters defined as on June 20, 1997. Once again, post test flight data is compared with simulation results incorporating wind levels measured. The results are summarized in Table VII, followed by data from the flight.

	Linear Simulation	Nonlinear Simulation	**Flight Test**
Lateral Mean Error	-2.419 feet	-2.431 feet	5.49feet
Lateral RMS Error	11.162feet	11.1869 feet	21.41 feet
Lateral Peak Error	15.24 feet	15.21 feet	41.5 feet
Longitudinal Mean Error	-.006 feet	-.005 feet	3.658 feet
Longitudinal RMS Error	.058 feet	.091 feet	14.949 feet
Longitudinal Peak Error	.089 feet	.0584 feet	37.1725 feet

Table VII. Error Comparison for Flight of July 23, 1997: Flight Test & Simulation

The resulting path in space flown on July 23 is shown in Figure 57. It is compared to the reference trajectory. The projection of the trajectory tracked onto the horizontal plane is shown in Figure 56. It, also, is compared to results from

simulation. The history of the controller activity is shown for the lateral channel in Figure 58, and for the longitudinal channel in Figure 59.

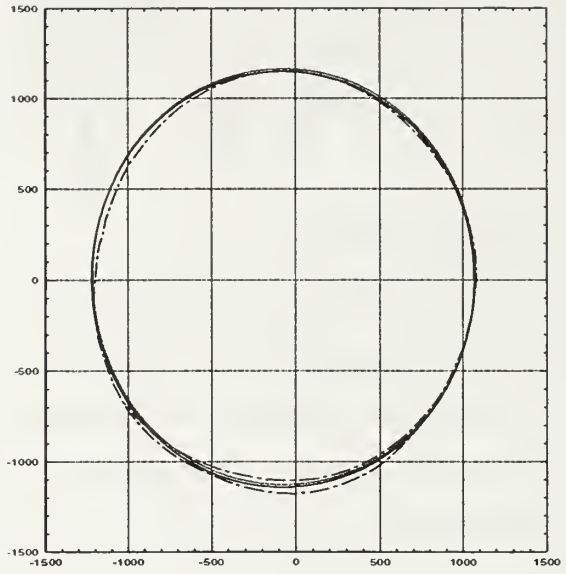


Figure 56. The second autonomous flight of *Frog*. Weather conditions were near ideal as *Frog* tracked the 3-D trajectory shown by the dashed line. The results from simulation are shown as a dash-dot line.

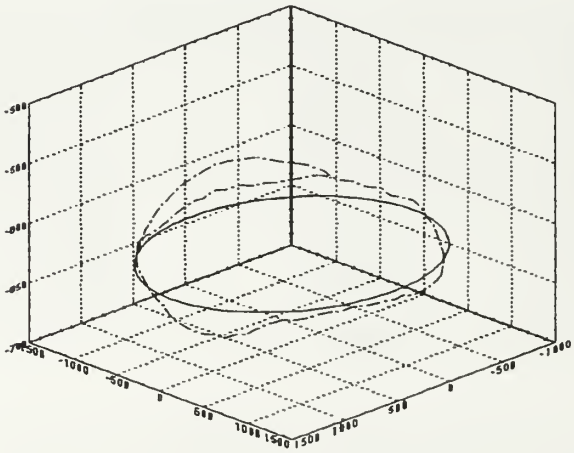


Figure 57. The second autonomous flight of *Frog*. Weather conditions were near ideal as *Frog* tracked the 3-D trajectory shown.

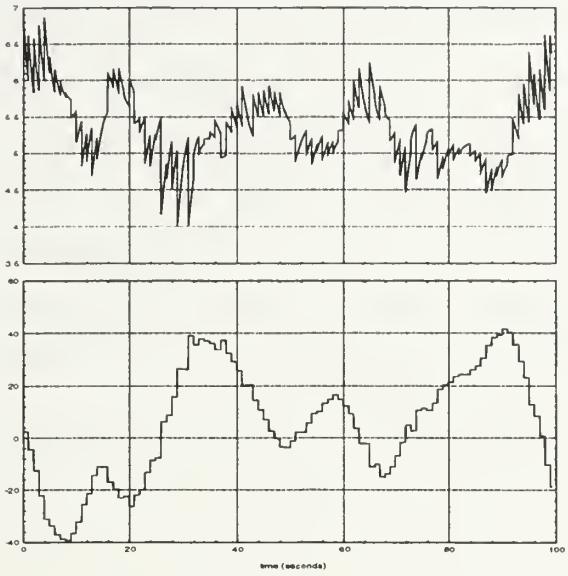


Figure 58. Top graph: Commanded vehicle yaw rate, in degrees per second from the controller, along the trajectory flown on July 23, 1997. Bottom graph: Lateral error in feet, as computed by the guidance algorithm, along the same trajectory.

With little wind, the test data characterizes the performance of the guidance and control algorithm in the presence of modeling errors, transport lag, hardware nonlinearities, and sensor noise. The simulation results were a close match to flight test data along the lateral channel. The average tracking error just over 21 feet. It can be seen in Figure 58 that the integral control is holding about one half degree per second commanded yaw rate to compensate for constant disturbances. In flight, tracking errors along the longitudinal channel were even less than lateral tracking errors. The average command signal along the longitudinal channel was -5 feet per second. This probably indicates that the power setting was too high. Subsequent testing will incorporate airspeed and throttle control which will address this issue. In simulation, however, longitudinal tracking errors were extremely small. The discrepancy is most likely the result of unmodeled vertical disturbances in the airmass due to light thermal activity.

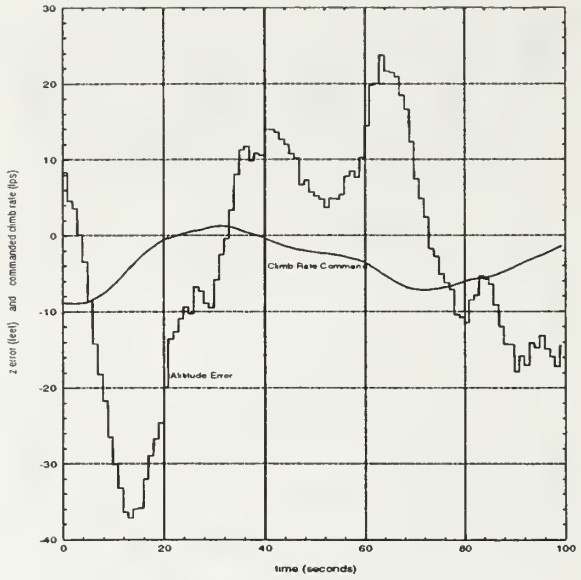


Figure 59. The output of the longitudinal channel of the controller is shown along with the error signal on which it is acting. Note that the positive z-axis points down and that positive error corresponds to the vehicle being below the reference trajectory.

F. CONCLUSIONS

In this chapter, an integrated system for the design, development and testing of guidance, navigation, and control algorithms for unmanned air vehicles was presented. Extensive use was made of commercial off-the-shelf (COTS) hardware. This kept costs low, and made the system easily scaleable. Sophisticated autocode tools allowed a two man team to write, test, and maintain thousands of lines of error free real time code. In a single, unified environment, the avionics system was designed, simulated, simulated incorporating hardware-in-the-loop (HITL), tested in flight, and used to control an aircraft in flight.

As a proof-of-concept demonstration, the RFTPS was used to take the integrated guidance and control concept presented in Chapter III and evaluate it in the environment for which it was proposed to be used. The project began with an unmanned air vehicle with unknown flight characteristics. In addition, a "black box" autopilot with unknown internal dynamics was placed onboard the vehicle. Through the use of the RFTPS, a high fidelity simulation of the vehicle and autopilot was built

and used to synthesize the applicable guidance and control laws. Extensive testing in simulation followed, and appropriate user interfaces were developed. Then, the RFTPS was used to control the vehicle in flight using the guidance and control laws developed, collect the data during the flight test, post process the data, and evaluate the performance of the algorithms.

The success of the project demonstrated both the utility of the integrated guidance and control algorithm as well as the capabilities of the RFTPS. The integrated guidance and control algorithm was shown to work well in a real world application. Performance in flight was very close to performance in simulation, which speaks well of the robustness properties of the controller. Changing environmental conditions highlighted some disturbance rejection issues that should be addressed in the future. The RFTPS was shown to be powerful, portable, rugged, effective in the field and in the lab, and safe and reliable at controlling an aircraft.

APPENDIX A. THE TAIL SIZING DESIGN TOOL

1. INTRODUCTION

This appendix details the operation of the *Tail Sizing Design Tool.* The *Tail Sizing Design Tool* runs from the MATLAB command line, and requires the LMI and CONTROL Toolboxes. The purpose of the *Tail Sizing Design Tool* is to allow the user to map and view the *Tail Sizing Design Space* for a given aircraft definition, or compare the *Tail Sizing Design Space* of two different aircraft definitions. There are three versions of the *Tail Sizing Design Tool.* The *Tail Sizing Design Tool* Version A maps the *Tail Sizing Design Space* for the high angle-of-attack excursion searching over static, state feedback, controllers. Version B solves the same problem but searches over dynamic, output feedback, controllers. Version C maps the *Tail Sizing Design Space* for the wind-shear penetration constraint searching over static state feedback controllers. Version A and B are well developed and allow the user to include aeroelastic effects, the addition of a canard, and to view the *Tail Sizing Design Space* in a three dimensional plot. Version C is less well developed and provides for only rigid body, tail only, aircraft dynamics, and two dimensional views.

The general structure of all three versions is similar. The *Tail Sizing Design Tool* is comprised of three directories which must reside on the same level within the user's file system. The first directory is termed *GUI*, which is an acaronym for graphical user interface. As the name suggests, it contains files which allow the user to specify an aircraft definition, generate a database for an aircraft definition or view a *Tail Sizing Design Space* for an aircraft definition by pushing the appropriate "buttons" on the display generated. The second directory is termed *MODEL*, and it contains the files necessary to model the dynamics of an aircraft. This directory contains the user supplied aerodynamic stability and control derivatives. The third directory is termed *DATA*, and it contains data files generated by running the *Tail*

Sizing Design Tool.

Assume for the moment that the proper stability and control derivatives have been placed appropriately in the *MODEL* directory. The process is as follows.

1. From the GUI directory, run *size_it.*
2. Fill in the appropriate values for the range of tail volumes and actuator rates of interest.
3. Specify the aircraft definition parameters such as the use of a canard or inclusion of a flexible mode.
4. For output feedback controllers, specify the output sensors to be used from the available choices.
5. Name the database file and start the generation of the data. This executes the program *getdata.*
6. Run the program *view_it* to querry a database previously generated.

2. GENERATING A DATABASE

First, we will consider the two programs *size_it* and *getdata.* The interface generated by *size_it* (Version B) is shown in Figure 60. By selecting the push button, "Output Page," another GUI is created (Figure 61). This provides the user the option of specifying which sensors to use for feedback. When the user is satisfied with the definition of the aircraft, he begins execution by selecting the pushbutton *Run It!.* This executes the program *Getdata*, shown next for Version B.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Getdata.m              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize global variables and move to the model directory.
cgrate=[]; data=[];
cd ..
cd model

%Use the graphics handles to transfer the user defined aircraft
%defintions to variables in the workspace. Begin with the frequency
```

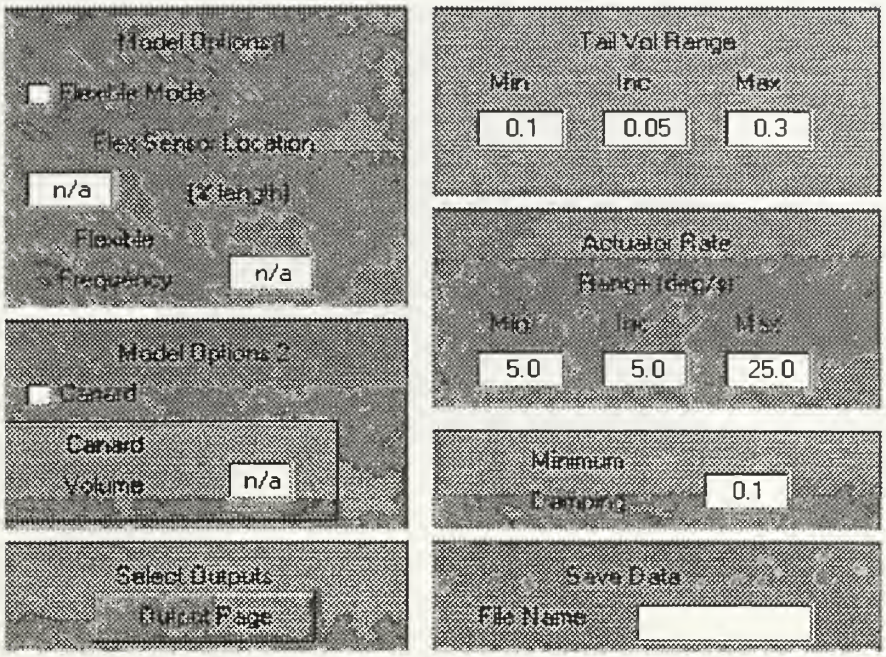


Figure 60. Interface called by size_it.m

```
% of the flexible mode.
freq1 = str2num(get(freq_ed,'string'));
save flexdata freq1
%This is the location of the flexible mode sensors.
sens_loc = str2num(get(sens_ed,'string'));
%Some locations (nodes/anti-nodes) make the problem
%unobservable. Check for this and alert the user if it is a problem.
[phi,phidot]=modeshpe(sens_loc%250/100);
if (phidot < 1e-5),
 disp('Flexible pitch rate is nearly unobservable.')
 disp('Change the sensor location ... stopping')
 clf
 % Message
  frame_title = uicontrol(gcf,...
'style','text',...
'position',[120 270 200 30],...
'string','There was a problem,',...
'foregroundcolor',[1 1 1],...
'backgroundcolor',[0 0 0]);
  % Message
  frame_title = uicontrol(gcf,...
```

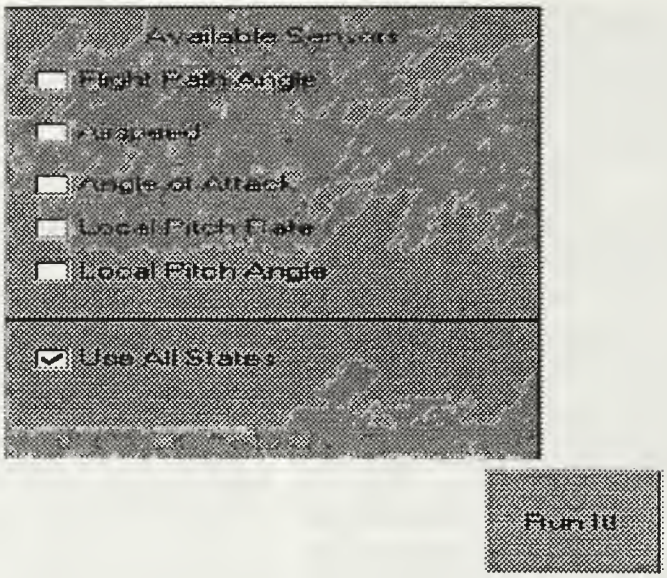


Figure 61. Selection of output sensors

```
'style','text',...
'position',[120 240 200 30],...
'string','see the command window.',...
'foregroundcolor',[1 1 1],...
'backgroundcolor',[0 0 0]);
  clear
  break;
end;
%Name the file to store the data.
opfname = get(save_ed,'string');
%Integer 1 if model is aeroelastic.
flexflag = get(ck_flex,'value');
%Integer 1 if the model has a canard.
canardflag = get(ck_canard,'value');
%If a canard is present, specify its canard volume.
if(canardflag==1),
split = str2num(get(canard_ed,'string'));
else, split = 0; end;
%This is the range and increments of tail volume to check.
vhlo = str2num(get(minvol_ed,'string'));
vhhi = str2num(get(maxvol_ed,'string'));
vhinc = str2num(get(incvol_ed,'string'));
%This is the required closed loop damping.
damping = str2num(get(damp_ed,'string'));
```

```
theta = 2%acos(damping); % pole placement region definition
%This is the range and increment of actuator rates.
ratelo = str2num(get(minrate_ed,'string'));
ratehi = str2num(get(maxrate_ed,'string'));
rateinc = str2num(get(increate_ed,'string'));
%This specifies which sensors to use for output feedback.
% An integer 1 indicates use of the sensor.  Choices are
% flight path angle, airspeed angle of attack, local
% pitch rate, local pitch angle.  The default is full
% information output feedback.
gamaflag = get(ck_gama,'value');
asflag = get(ck_as,'value');
aoaflag = get(ck_aoa,'value');
lprflag = get(ck_lpr,'value');
lpaflag = get(ck_lpa,'value');
fsflag = get(ck_fs,'value');

% The actuator dynamics are specified in the function
% ACTAUTOR.M. Canard and tail actuators do not have
% to be the same although dummy dynamics must be
% specified for a canard even if not used.
[Aa,Ba,Ca,Da] = actuator;
act=length(Aa);
% For the first order actuator modelused here,
% specify the time constant
a_bw = 20;

% The angle of attack excursion is equal to acos(gust/vt).
gust = 66; %fps

% The equilibrium point is specified by the flight
% path angle and true airspeed of the HSCT.
gamma = 0; vt = 422; % deg fps

% Specify tolerance limit for the cg binary search.
tol = 0.01;

% MATLAB uses the matrix variable ''region'' to define
% the closed loop poleplacement region.  Form it
% based on user inputs.
region=zeros(4,8);
region(1,1)=.02+1i;
```

```
region(2,2)=-40+1i;
region(3,3)=0+2i;
region(1,5)=1;
region(2,6)=-1;
region(3,7)=cos(damping);
region(4,8)=cos(damping);
region(3,8)=-sin(damping);
region(4,7)=sin(damping);

% Set up three loops to map the design space.
% First, loop through range of tail volumes.
for vh=vhlo:vhinc:vhhi;

% Initialize some variables for each tail volume.
yemax=0; ydemax=0; ycmax=0; ydcmax=0;

% Loop through the range of actuator rates.
for ratelimit = ratelo:rateinc:ratehi;

% Reset cg upper and lower bounds for the
% binary search.
xcgmax = 1.0; xcgmin = 0;


% Binary search on c.g. ;drop out when
% within tolerance.
while (norm(xcgmax-xcgmin) > tol);

save cgvh xcg vh split;

% Load appropriate data for trim; initial
% guesses for trim.
load trimdata

% Trim at cg specified and linearize at peak
% of gust condition.
if ( flexflag == 1),
[xtrim,utrim,ytrim] = trim('eom_b70w',x0,u0,....
[vt;gamma],[],[1],[1;2]);
xtrim(2) = xtrim(2) + gust;
[af,bf,cf,df] = linmod('eom_b70w',xtrim,utrim);
bf = bf(:,1:2); df = df(:,1:2);
```

```
xtrim(2) = xtrim(2) - gust;
else,
[xtrim,utrim,ytrim] = trim('eom_b70r',x0r,u0,....
[vt;gamma],[],[1],[1;2]);
xtrim(2) = xtrim(2) + gust;
[af,bf,cf,df] = linmod('eom_b70r',xtrim,utrim);
bf = bf(:,1:2); df = df(:,1:2);
xtrim(2) = xtrim(2) - gust;
end;

% Connect the actuators in series.
[A,B,C,D] = series(Aa,Ba,Ca,Da,af,bf,cf,df);
[m,n]=size(A);

% Place the actuator states after the plant states.
[n,p]=size(B);
T1 = [zeros(n-2,2) eye(n-2);eye(2) zeros(2,n-2)];
A = T1%A/T1;
B = T1%B;
C = C/T1;

% Similarity transformation used  to make local
% pitch rate and local body angle states vice
% using generalized coordinates.
if(flexflag==1),
[phi,phidot]=modeshpe(sens_loc%250/100);
T2 = [eye(4) zeros(4,2+act);
     [0 0 0 1 phidot 0] zeros(1,act);
     [0 0 1 0 0 phidot] zeros(1,act);
     zeros(act,6) eye(act)];
else, T2=eye(n); end;

Abar = T2A/T2;
Bbar = T2B;
Cbar = C/T2;

% Form output matrix based on user specified sensors.
% airspeed is Cbar(1,:)
% flight path angle is Cbar(2,:)
% angle of attack is Cbar(3,:)
% local pitch rate is 5th state [0 0 0 0 1 0 0 0]
% local pitch angle is 6th state [0 0 0 0 0 1 0 0]
```

```
C_select=[];
if gamaflag == 1;
 C_select = [C_select;Cbar(2,:)];
end;
if asflag == 1;
 C_select = [C_select;Cbar(1,:)];
end;
if aoaflag == 1;
 C_select = [C_select;Cbar(3,:)];
end;
if lprflag == 1;
 C_select = [C_select;[zeros(1,n-4) 1 0 0 0]];
end;
if lpaflag == 1;
 C_select = [C_select;[zeros(1,n-4) 0 1 0 0]];
end;
if fsflag == 1;
 C_select = eye(n);
end;
Cbar=C_select;
[n,m]=size(Bbar);
[p,n]=size(Cbar);

% Use the angle of attack excursion to define
% the initial condition.
i0=[0 gust 0 0 zeros(1,n-4-act) zeros(1,act)]';
%i0=T%i0;

% Specify rate and ampltude constraints.  The
% rate limit comes from the
% GUI. The saturation limit is set at 30 degrees.


ucmax = 30; uemax = 30;
udcmax = ratelimit; udemax = ratelimit


% Finally, form the LMIs.
setlmis([]);

R = lmivar(1, [n 1]);   %R%
S = lmivar(1, [n 1]);   % S %
```

```
Ak = lmivar(2, [n n]);  % Ak %
Bk = lmivar(2, [n p]);   % Bk %
Ck = lmivar(2, [m n]);   % Ck %

% This is pole placement LMIs formed using
% LMI region matrix REGION.
[rr,rc]=size(region);
if rc~=2%rr,
error('REGION must be a Mx(2M) matrix');
end
r1=0;
% scan each region (L,M) (diagonal blocks)
while r1<rr,
  bs=round(imag(region(r1+1,r1+1)));
  if ~bs, bs=rr-r1; end
  L=real(region(r1+1:r1+bs,r1+1:r1+bs));
  M=region(r1+1:r1+bs,rr+r1+1:rr+r1+bs);
  pole=newlmi;  % pole placement in the region (L,M)
  for i=1:bs,
    nbi=2%i-1;   % row of block in target LMI
    % off-diagonal 2x2  block
    for j=1:i-1,
      nbj=2%j-1; % col of block in target LMI
      lmiterm([pole,nbi,nbj,R],L(i,j),1);
      lmiterm([pole,nbi,nbj,R],Abar,M(i,j));
      lmiterm([pole,nbi,nbj,R],M(j,i),Abar');
      lmiterm([pole,nbi,nbj,Ck],Bbar,M(i,j));
      lmiterm([pole,nbi,nbj,-Ck],M(j,i),Bbar');
      lmiterm([pole,nbi+1,nbj+1,S],L(i,j),1);
      lmiterm([pole,nbi+1,nbj+1,S],M(i,j),Abar);
      lmiterm([pole,nbi+1,nbj+1,S],Abar',M(j,i));
      lmiterm([pole,nbi+1,nbj+1,Bk],M(i,j),Cbar);
      lmiterm([pole,nbi+1,nbj+1,-Bk],Cbar',M(j,i));
      lmiterm([pole,nbi+1,nbj,0],L(i,j));
      lmiterm([pole,nbi+1,nbj,Ak],M(i,j),1);
      lmiterm([pole,nbi+1,nbj,0],M(j,i)%Abar');
      lmiterm([pole,nbi,nbj+1,0],L(i,j));
      lmiterm([pole,nbi,nbj+1,-Ak],M(j,i),1);
      lmiterm([pole,nbi,nbj+1,0],M(i,j)%Abar);
     end
    % diagonal 2x2  block
    lmiterm([pole,nbi,nbi,R],L(i,i),1);
```

```
    lmiterm([pole,nbi,nbi,R],Abar,M(i,i),'s');
    lmiterm([pole,nbi,nbi,Ck],Bbar,M(i,i),'s');
    lmiterm([pole,nbi+1,nbi+1,S],L(i,i),1);
    lmiterm([pole,nbi+1,nbi+1,S],M(i,i),Abar,'s');
    lmiterm([pole,nbi+1,nbi+1,Bk],M(i,i),Cbar,'s');
    lmiterm([pole,nbi+1,nbi,0],L(i,i));
    lmiterm([pole,nbi+1,nbi,Ak],M(i,i),1);
    lmiterm([pole,nbi+1,nbi,0],M(i,i)%Abar');
  end
  r1=r1+bs;
end  % while


% This is x_0 LMI from the invariat elipsoid.
x0_lmi = newlmi;
lmiterm ( [-x0_lmi 1 1 0], 1);
lmiterm( [-x0_lmi 2 1 0], i0);
lmiterm( [-x0_lmi 2 2 R],1,1);
lmiterm( [-x0_lmi 3 1 S], 1, i0);
lmiterm( [-x0_lmi 3 2 0], eye(n));
lmiterm( [-x0_lmi 3 3 S], 1, 1);

% This is (u_max) saturation limit elevator LMI.
umaxe_lmi = newlmi;
lmiterm([-umaxe_lmi 1 1 R], 1,1);
lmiterm( [-umaxe_lmi 2 1 0],eye(n));
lmiterm([-umaxe_lmi 2 2 S], 1,1);
lmiterm([-umaxe_lmi 3 1 Ck], [0 1],1);
lmiterm([-umaxe_lmi 3 2 0], 0%eye(n));
lmiterm( [-umaxe_lmi 3 3 0], uemax^2);

% This is (u_max) saturation limit canard LMI.
if(canardflag==1),
umaxc_lmi = newlmi;
lmiterm([-umaxc_lmi 1 1 R], 1,1);
lmiterm( [-umaxc_lmi 2 1 0],eye(n));
lmiterm([-umaxc_lmi 2 2 S], 1,1);
lmiterm([-umaxc_lmi 3 1 Ck], [1 0],1);
lmiterm([-umaxc_lmi 3 2 0], 0%eye(n));
lmiterm( [-umaxc_lmi 3 3 0], ucmax^2);
end;
```

```
% This is (ud_max) actuator rate elevator LMI.
udmaxe_lmi = newlmi;
lmiterm([-udmaxe_lmi 1 1 R], 1,1);
lmiterm( [-udmaxe_lmi 2 1 0],eye(n));
lmiterm([-udmaxe_lmi 2 2 S], 1,1);
lmiterm([-udmaxe_lmi 3 1 Ck], [0 a_bw],1);
lmiterm([-udmaxe_lmi 3 1 R], [zeros(1,n-1) -a_bw],1);
lmiterm([-udmaxe_lmi 3 2 0], [zeros(1,n-1) -a_bw]);
lmiterm( [-udmaxe_lmi 3 3 0], udemax^2);

% This is (ud_max) actuator rate canard LMI.
if(canardflag==1),
udmaxc_lmi = newlmi;
lmiterm([-udmaxc_lmi 1 1 R], 1,1);
lmiterm( [-udmaxc_lmi 2 1 0],eye(n));
lmiterm([-udmaxc_lmi 2 2 S], 1,1);
lmiterm([-udmaxc_lmi 3 1 Ck], [a_bw 0],1);
lmiterm([-udmaxc_lmi 3 1 R], [zeros(1,n-2) -a_bw 0],1);
lmiterm([-udmaxc_lmi 3 2 0], [zeros(1,n-2) -a_bw 0]);
lmiterm( [-udmaxc_lmi 3 3 0], udcmax^2);
end;

lmisys = getlmis;

% Chech to see if the set of LMIs is feasible.
% TMIN should be negative for feasibility and
% XFEAS is the decision vector that validates
% the above LMIs.
[tmin,xfeas] = feasp(lmisys,[0 200 1e8 0 0]);

% Values of TMIN less than 0 are not strictly feasible
% but seem to be close enough to work.
if tmin < 1e-4;


% Recover the LMI variables.
R = dec2mat(lmisys,xfeas,R);
S = dec2mat(lmisys,xfeas,S);
ak = dec2mat(lmisys,xfeas,Ak);
bk = dec2mat(lmisys,xfeas,Bk);
ck = dec2mat(lmisys,xfeas,Ck);
```

```
% Construct the controller and form the close loop system.
[u,sd,v]=svd(eye(n)- R%S);  % factorize I-RS
sd=diag(sqrt(1./diag(sd)));
Ni=sd*v';  Mti=u*sd;
Ac=Ni*(ak-S*(Abar)R-bkCbarR-S*Bbar*ck)*Mti;
Bc=Ni*(bk);
Cc=(ck)*Mti;
P = ltisys(Abar,Bbar,Cbar,zeros(p,m));
cont = ltisys(Ac,Bc,Cc,zeros(m,p));
clsys = slft(P,cont,m,p);
[acl,bcl,ccl,dcl] = ltiss(clsys);

% Simulate the closed loop system with the specified
% initial condition and record the maximum actuator
% amplitudes and rates.
[ye,x,t] = initial(acl,[Bbar(:,2);zeros(n,1)],....
[zeros(1,n) [0 1]*Cc],0,[i0;zeros(n,1)]);
yemax = max(abs(ye));
[yc,x,t] = initial(acl,[Bbar(:,2);zeros(n,1)],....
[zeros(1,n) [1 0]*Cc],0,[i0;zeros(n,1)]);
ycmax = max(abs(yc));
[yde,x,t] = initial(acl,[Bbar(:,2);zeros(n,1)],....
[[zeros(1,n-1) -a_bw][0 a_bw]*Cc],0,[i0;zeros(n,1)]);
ydemax = max(abs(yde));
[ydc,x,t] = initial(acl,[Bbar(:,2);zeros(n,1)],....
[[zeros(1,n-2) -a_bw 0][a_bw 0]*Cc],0,[i0;zeros(n,1)]);
ydcmax = max(abs(ydc));
eigvalue = eig(acl)
data = [data; xcg ratelimit vh split ycmax yemax ydcmax....
ydemax tmin utrim' xtrim' eigvalue'];

% Move cg limit aft.
xcgmin = xcg;

% Otherwise, move cg limit forward,
else,
xcgmax = xcg;
end; %SET FEASIBLE, if

% Chech to see if aft cg limit is
% within tolerance
if ( abs(xcgmax-xcgmin) < tol | xcg > .95 | xcg < .05 )
```

```
break;
end;  %WITHIN TOLERANCE, if
end; %BINARY CG SEARCH, while

% This is the aft most cg for the rate, tail
% volume combination.
if(ydcmax ~= 0 | ydemax ~= 0),
cgrate=[cgrate; vh xcg ydcmax ycmax ydemax yemax eigvalue'];
end;
end; %RATE SWEEP, for
end; %VH SWEEP, for

% Change to the DATA directory and store the
% result with the name provided from the GUI.
% Return the user to the GUI directory.
cd ..
cd data
eval(['save ',opfname,'.mat cgrate data '])
cd ..
cd gui
```

The above code is associated with the output feedback problem. For the state feedback problem, the code is very similar. The essential difference is in the LMI formulation and reconstructionof the controller. That part of the code from Version A substantially different from Version B is shown next.

```
% Form the state feedback LMIs

setlmis([]);

Y = lmivar(1,[n 1]);
W = lmivar(2,[2 n]);

% This is Lyapunov lmi with real part constraint

lyap = newlmi;
lmiterm([lyap 1 1 Y], Abar+center*eye(n), 1, 's');
lmiterm([lyap 1 1 W], Bbar, 1, 's');

% This is the pole placement LMI
```

```
pplac = newlmi;
lmiterm([pplac 1 1 Y], sin(theta/2)*Abar, 1, 's');
lmiterm([pplac 1 1 W], sin(theta/2)*Bbar, M, 's');
lmiterm([pplac 2 1 Y], cos(theta/2)*Abar, 1);
lmiterm([pplac 2 1 W], cos(theta/2)*Bbar, M);
lmiterm([pplac 2 1 -Y], -cos(theta/2),Abar');
lmiterm([pplac 2 1 -W], -cos(theta/2)*M',Bbar');
lmiterm([pplac 2 2 Y], sin(theta/2)*Abar, 1, 's');
lmiterm([pplac 2 2 W], sin(theta/2)*Bbar, M, 's');

% This is the Y > 0 LMI

ypos = newlmi;
lmiterm([-ypos 1 1 Y],1,1);

% This is x_0 LMI

x0_lmi = newlmi;
lmiterm([-x0_lmi 1 1 0], 1);
lmiterm([-x0_lmi 2 1 0], i0);
lmiterm([-x0_lmi 2 2 Y],1,1);

% This is ue_max LMI

uemax_lmi = newlmi;
lmiterm([-uemax_lmi 1 1 Y], 1,1);
lmiterm([-uemax_lmi 2 1 W], C_amp_e,M);
lmiterm([-uemax_lmi 2 2 0], (uemax^2));

% This is ude_max LMI

udemax_lmi = newlmi;
lmiterm([-udemax_lmi 1 1 Y], eye(n),eye(n));
lmiterm([-udemax_lmi 2 1 W], C_rate_e*Bbar,M);
lmiterm([-udemax_lmi 2 1 Y], C_rate_e*Abar,eye(n));
lmiterm([-udemax_lmi 2 2 0], udemax^2);

if (canardflag == 1),
% This is uc_max LMI

ucmax_lmi = newlmi;
lmiterm([-ucmax_lmi 1 1 Y], 1,1);
```

```
lmiterm([-ucmax_lmi 2 1 W], C_amp_c,M);
lmiterm([-ucmax_lmi 2 2 0], (ucmax^2));

% This is udc_max LMI

udcmax_lmi = newlmi;
lmiterm([-udcmax_lmi 1 1 Y], eye(n),eye(n));
lmiterm([-udcmax_lmi 2 1 W], C_rate_c*Bbar,M);
lmiterm([-udcmax_lmi 2 1 Y], C_rate_c*Abar,eye(n));
lmiterm([-udcmax_lmi 2 2 0], udcmax^2);
end;

lmisys = getlmis;


[tmin,xfeas] = feasp(lmisys,[0 200 1e9 0 0]);

if tmin < 1e-4

% recover LMI variables

y = dec2mat(lmisys,xfeas,Y);
w = dec2mat(lmisys,xfeas,W);
Kbar = w*M/y;

% record initial condition responses

if(canardflag == 1),
yc = initial(Abar+Bbar*Kbar,Bbar,C_amp_c*....
Kbar,[0 0],i0);
ycmax = max(abs(yc));
yrc = initial(Abar+Bbar*Kbar,Bbar(:,1),....
C_rate_c*[Bbar*Kbar + Abar],0,i0);
ydcmax = max(abs(yrc));
end;

ye = initial(Abar+Bbar*Kbar,Bbar,C_amp_e*Kbar,[0 0],i0);
yemax = max(abs(ye));
yre = initial(Abar+Bbar*Kbar,Bbar(:,2),C_rate_e*....
[Bbar*Kbar + Abar],0,i0);
ydemax = max(abs(yre));
eigvalue = eig(Abar+Bbar*Kbar);
```

```
data = [data; xcg ratelimit vh split ycmax yemax ....
ydcmax ydemax tmin utrim' xtrim' Kbar(1,:) Kbar(2,:) eigvalue'];
```

Again, for the gust penetration/state feedback problem (Version C), the code is very similar. Principally, the formation of the linear model (which includes gust dynamics) is different. After that, the code proceeds in a similar manner to the angle of attack/state feedback problem (Version A). That part of the code from Version C substantially different from Version A is shown next.

```
% Gust penetration model formulation,
% including gust dynamics and
% LMI formulation.

% Load appropriate data for trim
load trimdata
[xtrim,utrim,ytrim] = trim('eom_b70r',x0r,u0,....
  [vt;gamma],[],[1],[1;2]);

% Linear model for given cg/vh

[af,bf,cf,df] = linmod('eom_b70r',xtrim,utrim);
bf = bf(:,2); df = df(:,2);


% Connect the actuators in series
[A,B,C,D] = series(Aa,Ba,Ca,Da,af,bf,cf,df);
C=[]; D=[];

% Place the actuator states after the plant states
[n,p]=size(B);
T = [zeros(n-1,1) eye(n-1);eye(1) zeros(1,n-1)];
A = T*A/T;
B = T*B;
C=eye(n);
D=zeros(n,p);

% Add gust states to end of model
```

```
Ag=zeros(n+2,n+2);
Ag(n+1,n+1)=lambda1;
Ag(n+2,n+2)=lambda2;
Ag(1:5,n+1)=-A(1:5,2);
Ag(1:5,n+2)=-A(1:5,2);
Abar1=[A zeros(n,2);zeros(2,n+2)] + Ag;
Bbar1=[B;0;0];
Cbar=eye(n+2);
Dbar=zeros(n+2,p);

T = eye(n+2);
T(2,n+1) = -1;
T(2,n+2) = -1;

Abar=T*Abar1/T;
Bbar=T*Bbar1;


% Specify Static Controller Structure

M = [eye(n) zeros(n,2); zeros(2,n+2)];

% Define gust i.c.

i0=[0 0 0 0 zeros(1,n-4-act) zeros(1,act) -gust gust]';

% Rate/Amplitude selection matrices

C_amp_e = 1;  C_rate_e = [zeros(1,n-act) 1];

% modify for gust states

C_rate_e = [C_rate_e 0 0];

% AOA output matrix

C_alpha = (57.3/vt)*[0 1 zeros(1,n-2) 0 0];

% Specify rate and amplitude constraints

uemax = amplimit;
udemax = ratelimit;
```

```
% form the LMIs

setlmis([]);

Y = lmivar(1,[n 1;2 1]);
W = lmivar(2,[1 n+2]);

% This is Lyapunov lmi with real part constraint

lyap = newlmi;
lmiterm([lyap 1 1 Y], Abar+center*eye(n+2), 1, 's');
lmiterm([lyap 1 1 W], Bbar, M, 's');

% This is the pole placement LMI

pplac = newlmi;
lmiterm([pplac 1 1 Y], sin(theta/2)*Abar, 1, 's');
lmiterm([pplac 1 1 W], sin(theta/2)*Bbar, M, 's');
lmiterm([pplac 2 1 Y], cos(theta/2)*Abar, 1);
lmiterm([pplac 2 1 W], cos(theta/2)*Bbar, M);
lmiterm([pplac 2 1 -Y], -cos(theta/2),Abar');
lmiterm([pplac 2 1 -W], -cos(theta/2)*M',Bbar');
lmiterm([pplac 2 2 Y], sin(theta/2)*Abar, 1, 's');
lmiterm([pplac 2 2 W], sin(theta/2)*Bbar, M, 's');

% This is the Y > 0 LMI

ypos = newlmi;
lmiterm([-ypos 1 1 Y],1,1);

% This is x_0 LMI

x0_lmi = newlmi;
lmiterm([-x0_lmi 1 1 0], 1);
lmiterm([-x0_lmi 2 1 0], i0);
lmiterm([-x0_lmi 2 2 Y],1,1);

% This is ue_max LMI

uemax_lmi = newlmi;
lmiterm([-uemax_lmi 1 1 Y], 1,1);
```

```
lmiterm([-uemax_lmi 2 1 W], C_amp_e,M);
lmiterm([-uemax_lmi 2 2 0], (uemax^2));

% This is alpha_max LMI

%amax_lmi = newlmi;
%lmiterm([-amax_lmi 1 1 Y], 1,1);
%lmiterm([-amax_lmi 2 1 Y], C_alpha,1);
%lmiterm([-amax_lmi 2 2 0], (alphalimit^2));

% This is ude_max LMI

udemax_lmi = newlmi;
lmiterm([-udemax_lmi 1 1 Y], eye(n+2),eye(n+2));
lmiterm([-udemax_lmi 2 1 W], C_rate_e*Bbar,M);
lmiterm([-udemax_lmi 2 1 Y], C_rate_e*Abar,eye(n+2));
lmiterm([-udemax_lmi 2 2 0], udemax^2);


lmisys = getlmis;

[tmin,xfeas] = feasp(lmisys,[0 200 1e9 0 0]);

if tmin < 1e-4

% recover LMI variables

y = dec2mat(lmisys,xfeas,Y);
w = dec2mat(lmisys,xfeas,W);
Kbar = w*M/y;

% record initial condition responses

ye = initial(Abar+Bbar*Kbar,Bbar,....
C_amp_e*Kbar,0,i0);
yemax = max(abs(ye));
yre = initial(Abar+Bbar*Kbar,Bbar,....
C_rate_e*[Bbar*Kbar + Abar],0,i0);
ydemax = max(abs(yre));

eigvalue = eig(Abar+Bbar*Kbar);
```

When the process has completed for the range of values specified for either Version A, B, or C, the user is returned to the MATLAB command prompt. The program results will have been stored in the *DATA* directory using the name supplied by the user. The user is free to view them or generate another data file.

3. VIEWING A DATABASE

After a database has been generated for a particular aircraft definiton, the program *view_it.m* is used to view the data. The program *view_it.m* presents the graphical user interface shown in Figure 62.

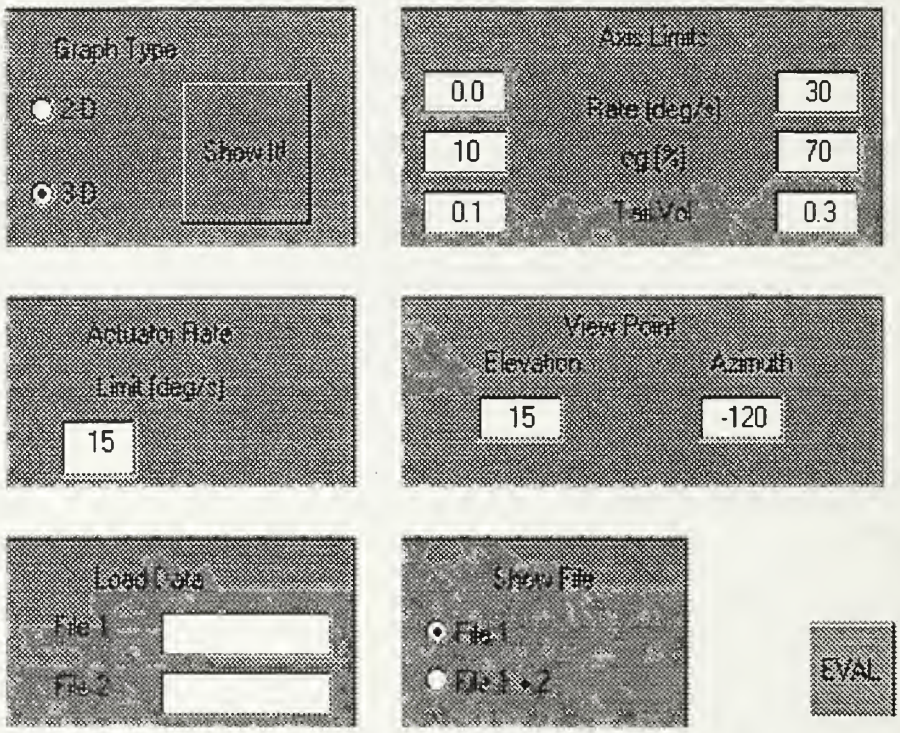


Figure 62. Interface called by view_it.m

The interface is fairly self explanatory. The options are to view the three dimensional Tail Sizing Design Space, or to view a 2D slice of the Tail Sizing Design Space. Additionally, the user can specify a second database, in order to compare two

different aircraft definitions on a single graph. If a two dimensional view is desired, then the user must specify the actuator rate limit used to intersect the lower surface of the Tail Sizing Design Space. When ready, the user selects the button labeled "Show It," which executes the program in the *GUI* directory called *showdata.m*.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Showdata.m               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
yi=[]; yi_e=[];

% Determine the user preferences and data file
% names from the GUI. Go load the file from
% the DATA directory.
ipfname1=get(load1_ed,'string');
ipfname2=get(load2_ed,'string');
cd ..
cd data
eval(['load ',ipfname1,'.mat']);
cgrateA=cgrate;
numfiles=1;
if ( get(rb_file2,'value') == 1 ),
 eval(['load ',ipfname2,'.mat']);
 cgrateB=cgrate;
 numfiles=numfiles+1;
end;
cd ..
cd model

% This is the target actuator rate: the level plane
% slicing the surface.
udselect=str2num(get(limit_ed,'string'));

% This records the user choice of either a 3D or 2D plot.
if ( get(rb_3d,'value') == 1)
 choice=1;
else,
 choice=0;
end;

% This captures the axis limits from the GUI.
udmin=str2num(get(minrate_ed,'string'));
```

```
udmax=str2num(get(maxrate_ed,'string'));
cgmin=str2num(get(mincg_ed,'string'))/100;
cgmax=str2num(get(maxcg_ed,'string'))/100;
vhmin=str2num(get(minvh_ed,'string'));
vhmax=str2num(get(maxvh_ed,'string'));

% This captures the view point preference.
azim=str2num(get(azim_ed,'string'));
elev=str2num(get(elev_ed,'string'));

% The user may wish to compare to aircraft definitions
% side by side. Loop for 1 or 2 files.
for comp=1:numfiles;
if (comp==1),
 cgrate=cgrateA;
else,
 cgrate=cgrateB;
end;

% Preprocess the data by parsing by tail volume.
[m,n]=size(cgrate);
num_vh=1;
index(num_vh)=1;
for i=1:m-1,
  if ( cgrate(i,1) < cgrate(i+1,1) ),
  index(num_vh+1)=i+1;
  num_vh=num_vh+1;
 end;
end;
index(num_vh+1)=m+1;
file='cgrate';
for i=1:num_vh,
 p=index(i);
 q=index(i+1)-1;
 eval([file,int2str(i),' = ',file,'(p:q,:);']);
end;

% Curve fit the rate vs. cg data for each tail
% volume.  The curve fit uses a simplex nonlinear
% fit. The general form of the function to fit
% to is two exponentials.
global DATA
```

```
file2='DATA';
file3='c';
lin=[]; nlin=[];
for k=1:num_vh;
 eval([file2,' = [',file,int2str(k),'(:,2) ',....
 file,int2str(k),'(:,5)]; '])
 lam = [1 0]';
 trace = 0;
 tol = .1;
 lambda = fmins('myfit',lam,[trace tol]);
 A = zeros(length(DATA(:,1)),length(lambda));
 for j = 1:length(lambda)
  A(:,j) = exp(-lambda(j)*DATA(:,1));
 end
 eval([file3,' = A\DATA(:,2);']);
 lin=[lin c];
 nlin=[nlin lambda];
end

% Repeat the process, this time curve fitting
% for elev amp vs. cg  for each tail volume.
global DATA_E
file2='DATA_E';
file3='c';
lin_e=[]; nlin_e=[];
for k=1:num_vh;
 eval([file2,' = [',file,int2str(k),'(:,2) ',....
 file,int2str(k),'(:,6)]; '])
 lam = [1 0]';
 trace = 0;
 tol = .1;
 lambda = fmins('myfit',lam,[trace tol]);
 A = zeros(length(DATA_E(:,1)),length(lambda));
 for j = 1:length(lambda)
  A(:,j) = exp(-lambda(j)*DATA_E(:,1));
 end
 eval([file3,' = A\DATA_E(:,2);']);
 lin_e=[lin_e c];
 nlin_e=[nlin_e lambda];
end

% Sample the functions determined above at uniform
```

```
% values of cg.
file4='vh';
t=cgmin:.01:cgmax;
t=t';
for z=1:num_vh;
 for j = 1:length(t)
  r(j,z) =  lin(1,z)*exp(-nlin(1,z)*t(j)) + ....
  lin(2,z)*exp(-nlin(2,z)*t(j));
  e(j,z) =  lin_e(1,z)*exp(-nlin_e(1,z)*t(j)) +
  lin_e(2,z)*exp(-nlin_e(2,z)*t(j));
 end;
 eval([file4,'(:,z) = ',file,int2str(z),'(1,1)*....
 ones(length(t),1); '])
end;

% This is the resulting mesh.  It could be plotted
% as is, but below we'll decrease the mesh size
% using linear interpolation.
ALLDATA=[];
ALLDATA_E=[];
for j=1:num_vh,
 ALLDATA=[ALLDATA t vh(:,j) r(:,j)];
 ALLDATA_E=[ALLDATA_E t vh(:,j) e(:,j)];
end;

% Linearly interpolate the mesh between the values of
% tail volume.
ht=[];
amp=[];
vol=[];
for k=1:((num_vh)-1);
 xi=vh(1,k):.01:(vh(1,k+1)-.01);
 x=vh(1,k:k+1);
 for j=1:length(t)
  y=r(j,k:k+1);
  y_e=e(j,k:k+1);
  yi(:,j)=interp1(x,y,xi);
  yi_e(:,j)=interp1(x,y_e,xi);
 end
 ht=[ht yi'];
 amp=[amp yi_e'];
 vol=[vol;xi'];
```

```
end
ht=[ht r(:,num_vh)];
amp=[amp e(:,num_vh)];
vol=[vol;vh(1,num_vh)];

% Plot the data.
if (comp==1),
 figure
end;
if (choice == 1),
 % This is the 3D plot of the data.
 htt=ht';
 [m,n]=size(htt);
 for p=1:m;
  for q=1:n;
   if (htt(p,q) > udmax | htt(p,q) < 0),
    ht(q,p)=nan;
   end;
  end;
 end;

 if (comp==1),
  surf(t*100,vol,ht')
  axis([cgmin*100 cgmax*100 vhmin vhmax udmin udmax])
  caxis([udmin udmax])
  grid;
  view(azim,elev)

  % This is the rate limit slice.
  ts=cgmin+.1:.01:cgmax-.1;
  [mm,nn]=size(ht);
  pln=ones(length(ts),nn)*udselect;
  C=ones(length(ts),nn)*2;
  surface(ts*100,vol,pln',C');
  xlabel('c.g. (%c)')
  ylabel('tail volume')
  zlabel('actuator rate (deg/s)')
 else,
   surf(t*100,vol,ht')
 end;

else,
```

```
% This is the 2D intersection of the target rate limit
% and surface.
 vhsc=[];
 cgsc=[];
 [m,n]=size(ALLDATA);
 for j=1:3:n
  xi = udselect;
  yi = spline(ALLDATA(:,j+2),ALLDATA(:,j),xi);
  vhsc=[vhsc ALLDATA(1,j+1)];
  cgsc=[cgsc yi];
 end

% Fit a curve to the points representing the 2D graph.
CGVH=[cgsc' vhsc'];
[p,s]=polyfit(CGVH(:,1),CGVH(:,2),2);
lam = [1 0]';
trace = 0;
tol = .1;
lambda = fmins('myfit',lam,[trace tol]);
A = zeros(length(CGVH(:,1)),length(lambda));
for j = 1:length(lambda)
A(:,j) = exp(-lambda(j)*CGVH(:,1));
end
c = A\CGVH(:,2);
lin_s=c;
nlin_s=lambda;

% Sample the fitted function at uniform cg increments.
vhsc=[];
cgsc=[];
t=cgmin:.05:cgmax;
t=t';
for j = 1:length(t)
 vhsc(j,1) =  lin_s(1,1)*exp(-nlin_s(1,1)*t(j)) +
 lin_s(2,1)*exp(-nlin_s(2,1)*t(j));
end;
cgsc=t*100;

% Plot the 2D intersection of the level plane and surface.
 if(comp==1),
  plot(cgsc,vhsc,'o',cgsc,vhsc);
  grid;
```

```
  axis([cgmin*100 cgmax*100 vhmin vhmax]);
  xlabel('c.g. (%c)')
  ylabel('tail volume')
 else,
  plot(cgsc,vhsc,'*',cgsc,vhsc);
  text(cgmax-.15,vhmin+.025,'* = 2nd File');
 end;
 end;

 if (numfiles==2),
  hold;
  clear yi yi_e r e cgrate
 end;

end;

% Return the user to the GUI directory.
cd ..
cd gui
```

4. THE DYNAMIC AIRCRAFT MODEL

The model used by the Tail Sizing Design Tool simulates the longitudinal motion of a conventional aircraft. The configuration may or may not include a canard. The upper shell of the dynamic model is either the Simulink file *eom_b70r.m*, for rigid-body dynamics, or the file *eom_b70w.m*, for flexible body dynamics. The *trim* and *linearize* routines supplied by MATLAB are use to determine the equilibrium point, and generate a linear model for the LMI problem formulation. The simulink shells above call the function *eomb70r*, or *eomb70w*, in order to compute the derivative of the models' states. The computation is based, in part, on the stability and control derivatives supplied by the user, which are retrieved via a nested function call explained later. The description of *eomb70w* follows, as the Simulink shell is self explanatory.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  function:  eomb70w.m      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% The vector passed to the function is the
% vehicles' states.
function accel = eomb70w(x)

% Retrieve the center of gravity , tail volume
% and canard volume parameters

load cgvh;

% Retrieve the normalized aircraft derivatives
% at the center of gravity and control volumes
% specified.  Retrieve the flexible mode frequency.

[rho,a,CD0,CL,CM,QF,omegai,mi,Ti,s,b,c,m,To,Iyy,lh,lc]=....
b70dataw(xcg,vh,split);
load flexdata

% Parse the vector passed to the function  into
% its components and use them to compute terms
% necessary to "denormalize" the force and moment
% coefficients.
dc = x(1); de = x(2); dtrt = x(3);
u = x(4); w = x(5); q = x(6);
theta = x(7); eta = x(8); etad = x(9);

% Calculate the total velocity, vt, and dynamic
% pressure gravity (assumed constant) and
% angle of attack.
vt                  = (u^2 + w^2)^.5;
qbar                = .5*rho*(vt^2);
g                   = 32.2;
alpha               = asin(w/vt);
norm1               = rho*vt^2*s/2;
norm2               = rho*vt*s*c/4;
norm3               = rho*vt^2*s*c/2;
norm4               = rho*vt*s*c^2/4;
norm5               = rho*vt^2*s*c/2;
norm6               = rho*vt*s*c^2/4;

% Based on the stability and control derivatives
% retrieved by the function {\em b70dataw.m}, the
% coefficients above and the vehicles' states,
```

```
% the total lift, drag, pitching moment and elastic
% generalized force on the vehicle is computed.

L = norm1*(CL(1) + CL(2)*alpha + CL(7)*de + CL(8)*dc + ....
CL(5)*eta) + norm2*(CL(3)*0 + CL(4)*q + CL(6)*etad);

Ln = L/(qbar*s);
Dn = (CD0 + 1.01*Ln^2/pi/a);

D  = Dn*qbar*s;

M = norm3*(CM(1) + CM(2)*alpha + CM(7)*de + CM(8)*dc + ....
CM(5)*eta) + norm4*(CM(3)*0 + CM(4)*q + CM(6)*etad);

Qeta = norm5*(QF(1) + QF(2)*alpha + QF(6)*de + QF(7)*dc ....
+ QF(4)*eta) + norm6*(QF(3)*q + QF(5)*etad);

% The lift, drag and gravity force are resolved in
% the body reference frame.  Then, the linear, angular
% and elastc accelerations are computed in the body
% reference frame.

udot = -q*w - g*sin(theta) + (-cos(alpha)*D +....
sin(alpha)*L + To*dtrt)/m;
wdot = q*u + g*cos(theta)  + (sin(alpha)*D - ....
cos(alpha)*L)/m;
qdot = M/Iyy;
etadd = -omegai(1)^2*eta - 2*omegai*Ti*etad  + Qeta/mi;
```

Notice that the prior program utilized a nested function call to retrieve the total vehicle stability and control derivatives. These derivates are supplied by the function *b70dataw*, which provide user supplied values of the wing-body, tail and canard contributions to lift and pitch. If a flexible mode is retained, the user must also specify the mode shape, *in vacuo* frequency, and generalized mass of the first bending mode. Presented below is data for the generic HSCT model termed Ref A. When the flexible mode is included, the model is termed Ref B.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function:   b70dataw.m    %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% To begin, the user must specify the physical
% description of the aircraft including the
% frequency and generalized mass associated
% with the flexible mode.

% parameter/value/units/description
G =  32.174;  % gravitational constant
g =  G; % fps^2
W =  480425; % lbs
m =  W/G; % slugs
Iyy =  2.181e7; % slug-ft^2
S =  6298;  % ft^2
c =  78.5;  % ft
cbar =  c; % c
b =  105;  % wing span
U =  422;  % fps
rho =  .0023769;  % density
Q =  .5*rho*U*U;  % lbf/ft*ft
a  =  1.751; % aspect ratio
lh =  85;  % tail moment arm to ac
lc =  85; % canard moment arm to ac
ih =  0; % tail incidence angle
To =  1000;  % nominal thrust lbs
cloc =  .05;  % canard location
tloc =  .95;  % aero center of tail location
vc =  split;  % canard volume
vh =  vh;  % tail volume
omegai = 6.29;  % flexible frequency
mi =  [500];  % generalized mass
Ti =  .02;  % flexible damping

% The stability and control derivatives of the
% canard, wing-body and tail are entered. Those listed
% below are for a generic HSCT "like" aircraft termed
% ref A.  Following the development of Wykes'method,
% the user can enter terms that residualize the effects
% of the elastic motion. In this case, only the rigid
% body derivatives have been entered. The  nomenclature
% uses either no extension or the extension (_r) to
% indicate that the term is a rigid body term.  The
% extension (_e) is used for flexible body terms.
```

```
% Finally, the extension (_eor) is used for terms that
% scale the rigid body terms due to flexible effects.


% Drag data
CD0 = 0.002;
CDU = 0.0;

% Lift data
clift_total = 0.161;
clift_vh_term = 0.801642;
clift_wb_af0_r = 0.04;
clift_dwb_af0_er = 0.0;
clift_wb_af_r = 0.0422;
clift_wb_af_eor = 1.0;
clift_wb_q_r = 0.0558;
clift_wb_q_eor = 1.0;
clift_wb_adf_r = 0.0;
clift_wb_adf_eor = 0.0;
clift_wb_n_e  = 0.0;
clift_wb_dq_e = 0.0;
clift_h_q_r = 0.123;
clift_h_q_eor = 1.0;
clift_h_adf_e = 0.0;
clift_h_n_e  = 0.0;
clift_h_dq_e  = 0.0;
clift_h_ah0_e  = 0.0;
clift_h_ah_r  = 0.0455;
clift_h_ah_eor  = 1.0;
clift_e_af0_r  = 1.8;
clift_e_af0_er  = 0.0;
clift_dah_dloadh = 0.0;
clift_e_af_r  = 0.71;
clift_e_af_eor = 1.0;

% Lift Data - Canard
clift_c_q_r  = 0.123;
clift_c_q_eor  = 1.0;
clift_c_adf_e = 0.0;
clift_c_n_e  = 0.0;
clift_c_dq_e  = 0.0;
clift_c_ac0_e  = 0.0;
```

```
clift_c_ac_r  = 0.0455;
clift_c_ac_eor  = 1.0;
clift_dac_dloadc = 0.0;
CLU = 0;

% Pitching moment data
cm_total  = -0.063618;
cm0_wb_r = 0.0;
cm0_wb_er  = 0.0;
dcm_dcl_wb_r  = 0.02;
dcm_dcl_wb_er = 0.0;
cm_wb_q_r = -0.0298;
cm_wb_q_eor = 1.0;
cm_wb_adf_r = -0.0067;
cm_wb_adf_eor = 1.0;
cm_wb_n_e =  -0.01;
cm_wb_dq_e = 0.0;
CMU = 0.0;

% Based on theses derivatives and the physical
% characteristics of the aircraft, some useful
% coefficients are calculated.

k1 = cbar/(2*U);
k2 = cbar*vh/lh;
k3 = S*cbar/lh;
k2_c = cbar*vc/lc;
k3_c = S*cbar/lc;

% Tail contribution coefficients

Alpha_1 = clift_h_ah0_e...
/(1-(clift_h_ah_r*clift_h_ah_eor*clift_dah_dloadh*Q*vh*k3));
Alpha_2 = -(clift_h_ah_r*clift_h_ah_eor*....
(clift_e_af0_r+clift_e_af0_er))...
/(1-(clift_h_ah_r*clift_h_ah_eor*clift_dah_dloadh*Q*vh*k3));
Alpha_0 = Alpha_1 + Alpha_2;
Alpha_ih= clift_h_ah_r*clift_h_ah_eor...
/(1-(clift_h_ah_r*clift_h_ah_eor*clift_dah_dloadh*Q*vh*k3));
Beta = (clift_h_ah_r*clift_h_ah_eor*....
(1-clift_e_af_r*clift_e_af_eor))...
/(1-(clift_h_ah_r*clift_h_ah_eor*clift_dah_dloadh*Q*vh*k3));
```

```
% Canard contribution coefficients.  Note, the
% canard is treated simply as a forward tail.

Alpha_0c= clift_c_ac0_e...
/(1-(clift_c_ac_r*clift_c_ac_eor*clift_dac_dloadc....
*Q*vc*k3_c));
Alpha_ic= clift_c_ac_r*clift_c_ac_eor...
/(1-(clift_c_ac_r*clift_c_ac_eor*clift_dac_dloadc....
*Q*vc*k3_c));
Beta_c = (clift_c_ac_r*clift_c_ac_eor*(1-clift_e_af_r....
*clift_e_af_eor))...
/(1-(clift_c_ac_r*clift_c_ac_eor*clift_dac_dloadc*Q*vc*k3_c));

% Finally, the total vehicle stability and control
% derivatives are computed based on the canard,
% wing-body and tail contributions.  The build up
% is based on a standard development presented in
% many aircraft dynamics texts; here I used Etkin.
% The computation of the elastic stability and control
% derivatives follow the development earlier in this text.

% Form static stability derivaties needed for trim.

CL0 = clift_wb_af0_r + clift_dwb_af0_er + ....
k2*Alpha_0 + k2_c*Alpha_0c;
CLA = 57.3*(clift_wb_af_r*clift_wb_af_eor  + ....
k2*Beta + k2_c*Beta_c);
CLIS = k2*Alpha_ih;
CLIC = k2_c*Alpha_ic;
CM0 = cm0_wb_r + cm0_wb_er + (dcm_dcl_wb_r + ....
dcm_dcl_wb_er)*(clift_wb_af0_r + clift_dwb_af0_er)....
- Alpha_0*vh +Alpha_0c*vc + CL0*(xcg-.25);
CMA = 57.3*((dcm_dcl_wb_r +
dcm_dcl_wb_er)*(clift_wb_af_r*clift_wb_af_eor)...
- Beta*vh + Beta_c*vc) + CLA*(xcg-.25);
CMIS = -Alpha_ih*vh + CLIS*(xcg-.25);
CMIC =  Alpha_ic*vc + CLIC*(xcg-.25);

% Compute dynamic stability derivatives needed
% for linearization.
```

```
CLQ = 57.3*(clift_wb_q_r*clift_wb_q_eor + ...
clift_h_q_r*clift_h_q_eor*k2 - clift_c_q_r*....
clift_c_q_eor*k2_c);
CLAD = 57.3*k1*(clift_wb_adf_r*clift_wb_adf_eor + ...
clift_h_adf_e*k2 + clift_c_adf_e*k2_c);
CMQ = 57.3*((dcm_dcl_wb_r + dcm_dcl_wb_er)*....
(clift_wb_q_r*clift_wb_q_eor) + cm_wb_q_r*cm_wb_q_eor -....
clift_h_q_r*clift_h_q_eor*vh
clift_c_q_r*clift_c_q_eor*vc) +CLQ*(xcg-.25);
CMAD = 57.3*((dcm_dcl_wb_r + dcm_dcl_wb_er)*....
(cm_wb_adf_r*cm_wb_adf_eor) ...
- clift_h_adf_e*vh - clift_c_adf_e*vc) + CLAD*(xcg-.25);

% Compute stability derivatives for the
% generalized elastic coordinates. When these are
% non-zero, the model is aeroelastic.  The resulting
% model is termed Ref B is Chapter 2.
[phi,phidot]=modeshpe(.25*250);
CLeta  = CLA*phidot;
CLetad = CLA*phi/U;
CMeta = CLA*(xcg-.25)*phidot;
CMetad = CLAD*phi*(xcg-.25)/U;
Q0 = -CL0*phi;
QA = -CLA*phi;
QQ = -CLQ*phi;
QETA = -CLA*phidot*phi;
QETAD = -CLAD*phi*phi/U;
[phi,phidot]=modeshpe(tloc*250);
QIS = -CLIS*phi;
[phi,phidot]=modeshpe(cloc*250);
QIC = -CLIC*phi;

% Combine the individual terms into a vector
% to be returned.
CL = [CL0 CLA CLAD CLQ CLeta CLetad CLIS CLIC];
CM = [CM0 CMA CMAD CMQ CMeta CMetad CMIS CMIC];
QF = [Q0 QA QQ QETA QETAD QIS QIC];
```

APPENDIX B. RFTPS SUPPORT

1. INTRODUCTION

This appendix documents the C-code unique to the RFTPS design that is not automatically generated via the autocode tools. The focus is on the serial module, which processes both the IMU and GPS data. More complete hardware descriptions of the I/O modules supported by the AC100 architecture can be found in the AC100 Model C30 Supplemental Reference Manual. Wiring diagrams and pin assignments are on file in the Avionics Lab.

2. SERIAL MODULE

The serial module contains two multi-mode, bi-directional, channels. The user interface to the device is through the file *user_ser.c*. Within *user_ser.c*, three functions must be defined. The first, get_SERIAL_parameters, is used to initialize the channels. The second, user_SERIAL_out, defines the output from each channel and the third, user_SERIAL_in, processes received data in the input buffer. Currently, the serial module is not used to send data. Some of the shell in *user_ser.c* is not shown. The documentation focuses on RFTPS specific code associated with processing the GPS and IMU data.

```
/*************************************************************
** File          : user_ser.c
** Project     : RFTPS
** Edit level : 4
**
** Abstract:   : File contains functions which the user
** must define to interface with IP-SERIAL device
** driver.
** The functions must be called
**
** get_SERIAL_parameters
** user_sample_SERIAL_in
** user_SERIAL_out
```

```
**
** Templates for the functions are provided.
**
** get_SERIAL_parameters
** Function sets the asynchronous communication
** parameters for the IP-SERIAL module.  Ring buffer
** sizes used to store received data must also be
** specified.
**
** user_SERIAL_out
** Function is called every scheduler interval.  The
** user is responsible for creating a byte stream from the
** models floating point outputs.  The user must ensure
** that the when writing these bytes to the output buffers
** that the buffers are not overflown.
**
** user_sample_SERIAL_in
** Function is called every sampling interval.  The
** user is responsible for filling the floating-point
** vector which is used as input to the model for
** the current sampling interval.
**
**
** Modifications:
** ------------------------------------------------------
**     Creation : 7-1--93  Henry Tominaga
**     Revised  : 8-23-93  Brent Roman
**     Revised  : 11-18-93 Steve Lynch
**     Revised  : 9-1-94   Steve Lynch
**     Revised  : 1-17-96  Eric Hallberg [New IMU]
**     Revised  : 3-15-96  Eric Hallberg [IMU Serial A
**                                     / GPS Serial B]
**     Revised  : 5-01-96  Eric Hallberg [IMU binary]
*/

#include <stddef.h>
#include <stdlib.h>
#include "sa_types.h"
#include "stdtypes.h"
#include "actypes.h"
#include "ioerrs.h"
#include "errcodes.h"
```

```
#include "iodefs.h"
#include "iodriver.h"
#include "ipserial.h"
#include "printx.h"

#define NULL 0

  /* semaphores and serial parameters for each
     physical channel */
private struct
{ ISI_BOOLEAN  allSent; ISI_BOOLEAN  broken;
  unsigned baud;} line_state[2];

struct user_type
{
    int update_interval;
    int update_count;

};


typedef struct _bytes
{
 unsigned byte1 :8;
 unsigned byte2 :8;
 unsigned byte3 :8;
 unsigned byte4 :8;
}_bytes;

typedef union float_char
{
  float  fl;
  _bytes ch;
} float_char;

/* global variables used in user_ser_in ch A
   IMU data processing */

u_int buffer_data[40];
int index;
int tick = 0;
int first_frame_a = 0;
```

```
float last_float_a[14];
float last_float_a_1[14];
float last_float_a_2[14];
float last_float_a_3[14];
int    missed_cr = 0;


/* global variables used in user_ser_in ch B
   GPS processing */

int first_frame_b = 0;
float last_float_b[50], sol=299792458.0, l1f0=1575420000.0, k2;
unsigned long  icpold1, icpold2, icpold3, icpold4,
               icpold5, icpold6;
int num_bytes_old=0;

/* Function: get_SERIAL_parameters ++++++++++++++++++++
** Returns:
**     NONE
*/
public void get_SERIAL_parameters
(
unsigned int                         hardware_channel,
volatile struct user_param          *device_param,
volatile struct ring_buffer_param *rec_buffer,
IOdevice                            *device)
{
   struct user_type *user_ptr;
   serial_param_type *serptr;

   serptr = device->parameters;

   if (SERIAL_USER_PTR == NULL)
   {
SERIAL_USER_PTR = (void *)malloc(sizeof(struct user_type));
user_ptr = (struct user_type *)SERIAL_USER_PTR;
user_ptr->update_interval  = 1000;
user_ptr->update_count     = 0;
   }


/* Both IMU and GPS sensors send their data at
```

```
   9600 baud/RS-232 standard */
if (hardware_channel == chanA || hardware_channel == chanB)
{
/* set parameters for channel A (IP-SERIAL channel 1)*/
device_param->parity             = NONE;
device_param->baud_rate          = 9600;
device_param->stop_bits          = ONE;
device_param->transmit_data_size = 8;
device_param->receive_data_size  = 8;
device_param->clock_multiplier   = 16;
/* Set size for receive ring buffer.
   Here it is set to 10x the message size */
rec_buffer->buffer_size          = 600;
   }
   else
   {
      printx("INVALID CHANNEL\n");
   }
} /* get_SERIAL_parameters */


/* Function: user_SERIAL_out ++++++++++++++++++++++
*/
public RetCode user_SERIAL_out(IOdevice *device,
       float model_float[],
       u_int ser_channel)
{
   Byte         cbuffer[20];
   u_int        i;
   float_char   data;
   RetCode      return_val;

   return_val = OK;
   /*********************************************
   *  Given floating point model output,
   *  create  buffer which contains bytes
   *  to be transmitted across
   *  serial channel.
   *********************************************/
   if (numbytes_in_buffer(device->parameters) == 0)
   {
      for (i = 0; i < 5; i++)
```

```
      {
 data.fl = model_float[i];
 cbuffer[0] =(Byte) data.ch.byte1;
 cbuffer[1] =(Byte) data.ch.byte2;
 cbuffer[2] =(Byte) data.ch.byte3;
 cbuffer[3] =(Byte) data.ch.byte4;

 /************************************************
 * Fill the output buffer with data to be transmitted
 * by the background portion of the serial driver
 ************************************************/
return_val = write_serial(device->parameters,cbuffer,4);
if (return_val == -1)
{
printx("Error: Buffer overflow in User_ser on
        output to serial\n");
         }
      }
   }
   return OK;
} /* user_SERIAL_out */

/* Function: user_sample_SERIAL_in  +++++++++++++++++
*/
public RetCode user_sample_SERIAL_in(IOdevice *device,
float     model_float[],
u_int     ser_channel)
{
/**********READING CHANNEL A ****** IMU ********/
if (ser_channel==chanA)
{
/* Create a software buffer to hold receive
   buffer data  */
 u_int soft_buffer[600];
/* Declare varibiables needed */
  unsigned  char item[1];
   float k;
   int y,z,j;
   int s = 0;
   int p = 0;
   u_int w_high;
   u_int w_low;
```

```
   struct user_type *user_ptr;

/* The IMU data is sent as twos complement
   and must be scaled*/
   float   scale[] = {180/8191.0, -2.0/8191.0,
                      -2.0/8191.0, -2.0/8191.0,
    200.0/8191.0, 200.0/8191.0,
                    200.0/8191.0, 180.0/8191,
                    180.0/8191.0, 10.0/8191.0,
                     10.0/8191, 10.0/8191.0,
                      10.0/8191.0, 10.0/8191.0},
default_data[] = { 10.0, 10.0, -10.0, 10.0, 10.0, 10.0,
            10.0, 10.0, 10.0, 1.0, 1.0, 1.0, 1.0, 1.0};

   /*********************************************
   * set user pointer to buffer allocated
   * by get parameters
   * this buffer is passed around with the
   * structure device
   * and should only be accessed via the SERIAL_USER_PTR
   *
   *********************************************/
   user_ptr        = SERIAL_USER_PTR;

/* The first time the serial port is read;
   load default data */
if (first_frame_a==0){
for (j=0;j<14;j++){
last_float_a[j] = default_data[j];
last_float_a_1[j] = default_data[j];
last_float_a_2[j] = default_data[j];
last_float_a_3[j] = default_data[j];
}/* end if*/

model_float = last_float_a;
index = 0;
first_frame_a = 1;
return OK;
}/* end if*/

/* Process hardware buffer by moving data into
```

```
   the software buffer */
while( (numbytes_in_buffer(device->parameters)) > 1  ){
read_serial(device->parameters,1,item);
soft_buffer[s]=(u_int) item[0];
s=s+1;
} /* end while */

/* The IMU data string ends with a carriage return.
   Always sync on the carriage return*/
while ( p < (s)){
if (soft_buffer[p] == '\r'){
missed_cr = 1;
index = 0;
}/* end if*/
else{
if (missed_cr == 1){
buffer_data[ index ] = soft_buffer[p];
}/* end if */

index=index + 1;
if (index == 30){
index = 0;
missed_cr = 0;
}/*end if*/
}/* end else*/
p=p+1;
} /* end while */

/* Implement some rudimentary error detection to remove
* blatantly bad data from the IMU.  Since the IMU does not
* send a check sum, use multiple buffers to compare present
* with past values.  Remove data readings that are not
* physically possible.  Send last good data.  */

tick = tick + 1;
if (tick == 4)
tick = 1;

if (tick == 1)
{
for ( y=0; y<14; y++){
```

```
/* The number is in binary, 2's complement.
   Convert to decimal and scale */
  w_high = buffer_data[y*2];
   w_low =  buffer_data[y*2+1];
  k = 128*(w_high & 0x7F) + (w_low & 0x7F);

if (k > 8191)
k = k - 16384;

     last_float_a_1[y] = k*scale[y];

if (abs(last_float_a_1[y] - last_float_a_3[y]) < .1)
last_float_a[y] = last_float_a_1[y];

} /* end for loop */
}/*end if*/

if (tick == 2)
{
for ( y=0; y<14; y++){

  w_high = buffer_data[y*2];
   w_low =  buffer_data[y*2+1];
  k = 128*(w_high & 0x7F) + (w_low & 0x7F);

if (k > 8191)
k = k - 16384;

     last_float_a_2[y] = k*scale[y];

if (abs(last_float_a_2[y] - last_float_a_1[y]) < .1)
last_float_a[y] = last_float_a_2[y];

} /* end for loop */
}/*end if*/

if (tick == 3)
{
for ( y=0; y<14; y++){

  w_high = buffer_data[y*2];
   w_low =  buffer_data[y*2+1];
```

```
  k = 128*(w_high & 0x7F) + (w_low & 0x7F);

if (k > 8191)
k = k - 16384;

     last_float_a_3[y] = k*scale[y];

if (abs(last_float_a_3[y] - last_float_a_2[y]) < .1)
last_float_a[y] = last_float_a_3[y];

} /* end for loop */
}/*end if*/

/* Pass the IMU data back up.
for (j=0;j<14;j++){
model_float[j] = last_float_a[j];
}/* end for*/
}/* end if ch a */

/****** READING CHANNEL B**** GPS ********/

if (ser_channel==chanB){

unsigned char itemb[1], msg1[600], checksum1[1],
checksum2[1], mychecksum1,mychecksum2, status,
mode1, mode2, mode3, mode4, mode5, mode6,
msg3[45], checksum3[1], mychecksum3, msg4[62],
checksum4[1], mychecksum4;

struct user_type *user_ptr;

unsigned long  i, j, icp1, icp2, icp3,
icp4, icp5, icp6,
eph1, eph2, eph3, eph4,
eph5, eph6;

long   lat, lon, vel, heigps, heimsl;

int  num1, num2, num3, sat1, sat2,
sat3, sat4, sat5, sat6,
satt1, satt2, satt3, satt4,
satt5, satt6, num4,
```

```
velnor, velup, veleast;

float  psrng1, psrng2,  psrng3,  psrng4,
psrng5, psrng6, psrngrate1, psrngrate2,
psrngrate3, psrngrate4, psrngrate5,
psrngrate6, loctime, sattime1,sattime2,
sattime3, sattime4, sattime5, sattime6,
ecefx, ecefy, ecefz, velnor1, veleast1,
velup1, hd, timeref, pscor1, pscor2,
pscor3, pscor4, pscor5, pscor6,
psrcor1, psrcor2, psrcor3, psrcor4,
psrcor5, psrcor6, locsec, locsecf,
satsec1, satsec2, satsec3, satsec4,
satsec5, satsec6,satsecf1, satsecf2,
satsecf3, satsecf4, satsecf5,
satsecf6, heigps1, heimsl1,
vel1, lat1, lon1, ecefx1, ecefy1, ecefz1;

/*********************************************************
* set user pointer to buffer allocated by get parameters  *
* this buffer is passed around with the structure device  *
* and should only be accessed via the SERIAL_USER_PTR     *
* define                                                  *
*********************************************************/

   user_ptr        = SERIAL_USER_PTR;

/*********************************************************
* set user pointer to buffer allocated by get parameters  *
* this buffer is passed around with the structure device  *
* and should only be accessed via the SERIAL_USER_PTR     *
* define                                                  *
 *********************************************************/

/* If first time read, load default data */
if(first_frame_b==0){
for(i=0;i<50;i++){
last_float_b[i]=7.0;
first_frame_b=1;
icpold1=0.0; icpold2=0.0; icpold3=0.0; icpold4=0.0;
icpold5=0.0; icpold6=0.0;
}/*end for*/
```

```
}/*end if*/

/* The message is 61 bytes long and comes in once a second.
* Wait for the whole message to come in. */
num1 = (numbytes_in_buffer(device->parameters));
if ((num1 > 60) & (num1 == num_bytes_old) ){

for(i=0;i<num1;i++){
read_serial(device->parameters,1,itemb);
msg1[i]=itemb[0];
}/* end for */

}/*end if message in*/

/* The message is in ASCII Hex and starts with @@BA.
   Search for start of string. */
for (i=4;i<num1;i++){

if (msg1[i-4]=='@' & msg1[i-3]=='@' &
    msg1[i-2]=='B' & msg1[i-1]=='a' ){

checksum1[0] = msg1[i+61];

/*Convert message bytes to decimal numbers*/

lat = msg1[i+11]*0x1000000 + msg1[i+12]*0x10000 +
      msg1[i+13]*0x100 + msg1[i+14];

lon = msg1[i+15]*0x1000000 + msg1[i+16]*0x10000 +
      msg1[i+17]*0x100 + msg1[i+18];

heigps =msg1[i+19]*0x1000000 +msg1[i+20]*0x10000 +
       msg1[i+21]*0x100 + msg1[i+22];

heimsl = msg1[i+23]*0x1000000 +msg1[i+24]*0x10000 +
         msg1[i+25]*0x100 +msg1[i+26];

vel = msg1[i+27]*0x100 + msg1[i+28];

hd = msg1[i+29]*0x100 + msg1[30];

hd = hd/10.0;
```

```
status = msg1[i+60];

vel1=vel/100.0;  heigps1=heigps/100.0;  heimsl1=heimsl/100.0;

lat1=lat/100.0; lon1=lon/100.0;

/*GPS uses a check sum. Check if message is correct*/

mychecksum1='B'^'a';

for(j=0;j<61;j++){

mychecksum1^=msg1[i+j];

}/*end for*/
/* If its correct, update GPS data.
if (mychecksum1 == checksum1[0]){

last_float_b[0]=lat1;
last_float_b[1]=lon1;
last_float_b[2]=heigps1;
last_float_b[3]=heimsl1;
last_float_b[4]=vel1;
last_float_b[5]=hd;
last_float_b[6]=status;
last_float_b[7]=checksum1[0];
last_float_b[8]=mychecksum1;

}/*end if checksum ok*/

i=i+60;

}/* end if @@Ba */
/*  Pass the GPS data back up */
for (j=0;j<8;j++){
model_float[j] = last_float_a[j];
}/* end for*/

}/* end if ch b */
return OK;
} /* user_sample_SERIAL_in */
```

LIST OF REFERENCES

[1] I. Ashkenas, R. Magdaleno, D. McRuer, "Flight Control and Analysis Methods for Studying Flying and Ride Qualities of Flexible Transport Aircraft," NASA CR 172201, August, 1983.

[2] H. Ashley, *Engineering Analysis of Flight Vehicles*, Addison-Wesley, Reading, MA, 1974.

[3] Baxandall, P., and Liebeck, H., *Vector Calculus*, Clarendon Press, Oxford, 1986.

[4] Beaufrere, H., "Limitations of Statically Unstable Aircraft due to the Effects of Sensor Noise, Turbulence, and Structural Dynamics," Guidance, Navigation and Control Conference, Williamsburg, VA, August 18-20, 1986, pp. 721-731. AIAA PAPER 86-2203.

[5] Beaufrere, H., Soeder, S., "Longitudinal Control Requirements for Statically Unstable Aircraft," NAECON 1986; Proceedings of the National Aerospace and Electronics Conference, Dayton, OH, May 19-23, 1986. Volume 2 (A87-16726 05-01). New, York, Institute of Electrical and Electronics Engineers, 1986, pp. 421-433.

[6] S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM, Phila., PA, 1994.

[7] Chilali, M., and Gahinet., P.," $\mathcal{H}_\infty$ Design with Pole Placement Constraints", to appear in *IEEE Transactions on Automatic Control*, 1996.

[8] M. Chilali, P. Gahinet, C. Scherer, "Multi-Objective Output-Feedback Control Via LMI Optimization," IFAC Procedings, 1996, pp. 249-254.

[9] Costello, D., Kaminer, I., Carder, K., and Howard, R., "The Use of Unmanned Vehicle Systems for Coastal Ocean Surveys: Scenarios for Joint Underwater and Air Vehicle Missions," *Proceedings 1995 Workshop on Intelligent Control of Autonomous Vehicles*, Lisbon, Portugal, March 1995.

[10] Craig, J. J., *Robotics*, Addison-Wesley, 1989.

[11] Douglas, I., and Koehler, K., "Low-Cost Aircraft Navigation System to Aid Global Climate Change Studies," *NASANews@mercury.hq.nasa.gov*, RELEASE: 96-99, May 14, 1996.

[12] Doyle, J., Glover, K., Khargonekar, P., and Francis, B., "State space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems," *IEEE Transactions on Automatic Control.* Vol. AC- 34, No. 8, 1989, pp. 831–847.

[13] Elgersma, M., *Control of Nonlinear Systems Using Partial Dynamic Inversion*, Ph.D. Thesis, University of Minnesota, Minneapolis, MN, 1988.

[14] B. Etkin, L. Reid, *Dynamics of Flight, Stability and Control*, John Wiley and Sons, Inc., New York, NY, 1996.

[15] Fryxell, D., Oliveira, P., Pascoal, A., Silvestre, C., and Kaminer, I., "Navigation, Guidance and Control of AUVs: An Application to the MARIUS Vehicle," *IFAC Journal of Control Engineering Practice*, Vol. 4, No. 3, 1996, pp. 401-409.

[16] P. Gahinet and A. Nemirovskii , " General Purpose LMI Solvers with Benchmarks," *Proc. 32nd IEEE Conference on Decision and Control*, pp. 3162-3165, San Antonio, TX, December 1993.

[17] Gahinet, P., Nemirovski, A., Laub., A.J., Chilali, M., *LMI Control Toolbox*, The Mathworks Inc, May 1995.

[18] Gill,P.E., Murray, W. and Wright, M.H., *Practical Optimization*, Academic Press Inc., N.Y., N.Y., 1981, pp. 105-115.

[19] Godhavn, J. M., "Nonlinear Tracking of Underactuated Surface Vessels", *preprint*, 1996.

[20] Healey, A., and Lienard, D., "Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles," *IEEE Journal of Oceanic Engineering,* Vol. 18, No.3, pp. 327-339, 1993.

[21] F.M. Hoblit, *Gust Loads on Aircraft: Concepts and Applications*, AIAA Education Series, American Institute of Aeronautics and Astronautics, Washington, DC, 1988, pp. 8-9.

[22] R. Horn, C. Johnson, *Matrix Analysis*, Cambridge University Press, New York, NY, 1985.

[23] Integrated Systems, Inc., "Xmath Interactive System Identification Module", Part Number 000-0027-001, June 1994.

[24] Kaminer, I., *Motion Control of Rigid Bodies Using $\mathcal{H}_\infty$ Synthesis and Related Theory*, Ph.D. Thesis, University of Michigan, Ann Arbor, MI, 1992.

[25] Kaminer, I., Hallberg, E., Pascoal, A., and Silvestre, C., "On the Design and Implementation of a Trajectory Tracking Controller for a Fixed Wing Unmanned Air Vehicle," *Proceedings 1995 American Control Conference*, Seattle, June, 1995.

[26] I. Kaminer, R. Howard and C. Buttrill, "On the Development of Closed Loop Tail Sizing Criteria for an HSCT," To appear in Journal of Guidance, Control and Dynamics.

[27] Kaminer, I., Pascoal, A., Khargonekar, P., and Coleman, E., "A Velocity Algorithm for the Implementation of Gain-Scheduled Controllers," *Automatica.* Vol. 3, 1995, pp. 1185-1191.

[28] J. Kay *et al*, " Control Authority Issues in Aircraft Conceptual Design: Critical Conditions, Estimation Methodology, Spreadsheet Asessment, Trim and Bibliography," VPI-Aero-200.

[29] Lin, C. *Modern Navigation, Guidance, and Control Processing*, Prentice-Hall, 1991.

[30] E. Livne et al., "Lessons from Application of Equivalent Plate Structural Modeling to an HSCT Wing," Journal of Aircraft, Vol. 31, No. 4, July-Aug. 1994, pp. 953-960.

[31] A. Messac and P.D. Hattis, "Physical Programming Design Optimization for High Speed Civil Transport," Journal of Aircraft, Vol. 33, No. 2, March-April 1996, pp. 446-449.

[32] A. Nemirovskii and P. Gahinet, "The Projective Method for Solving Linear Matrix Inequalities ," *Proc. 1994 American Control Conference*, pp. 840-844, Baltimore, MD, June 1994.

[33] R. J. Niewhoener and I. Kaminer, "On Integrated Aircraft/Controller Design Using Linear Matrix Inequalities," *Journal of Guidance, Control and Dynamics*, Vol. 19, No. 2, March-April 1996, pp. 445-453.

[34] Papageorgiou, Evangelos, "Development of a Dynamic Model for a UAV", Master's Thesis, Naval Postgraduate School, March 1997.

[35] Papoulias, F.," Stability Considerations of Guidance and Control Laws for Autonomous Underwater Vehicles in the Horizontal Plane," *Proceedings 7th International Symposium on Unmanned Untethered Vehicle Technology*, Durham, N.H., 1991.

[36] Pascoal, A. M.," The AUV MARIUS: Mission Scenarios, Vehicle Design, Construction and Testing," *Proceedings 2nd Workshop on Mobile Robots for Subsea Environments*, Monterey, California, May 1994.

[37] J.K. Ray, C.M. Carlin, and A.A. Lambregts, "High-Speed Civil Transport Flight- and Propulsion-Control Technologies Issues," NASA CR 186015, March 1992.

[38] Roskam, J., *Airplane Flight Dynamics and Automatic Flight Controls,* Roskam Aviation and Engineering Corp., Ottawa, KS, 1979.

[39] Rugh, W. J., "Analytical framework for gain scheduling," *IEEE Control Systems Magazine*, vol. 11, No. 1, 1991, pp. 74-84.

[40] Samson, C., " Path Following and Time-Varying Feedback Stabilization of a Wheeled Mobile Robot," *Proceedings International Conference ICARCV'92*, RO-13.1, Singapore, 1992.

[41] Schmidt, D.K, "On the Integrated Control of Flexible Supersonic Transport Aircraft", Guidance, Navigation and Control Conference, Baltimore, MD, August 7-10, 1995, p. 258-265. AIAA-95-3200-CP.

[42] Silvestre, C., Pascoal, A. M., Fryxell, D., and Kaminer, I., "Design and Implementation of a Trajectory Tracking Controller for an Autonomous Underwater Vehicle," *Proceedings 1995 American Control Conference*, Seattle, June, 1995.

[43] Slattery, R. A., and Zhao, Y., "En-Route Descent Trajectory Synthesis for Air Traffic Control Automation," *Proceedings 1995 American Control Conference*, Seattle, June 1995.

[44] Walsh, G., Tilbury, D., Sastry, S., Murray, R., and Laumond, J. P., "Stabilization of Trajectories for Systems with Nonholonomic Constraints," *IEEE Transactions on Automatic Control*, Vol. 39, No 1, 1994, pp. 216-222.

[45] M. Waszak, D. Schmidt, "Flight Dynamics of Aeroelastic Vehicles," Journal of Aircraft, VOL. 25, NO. 6, June 1988, pp. 563-571.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road.,Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Dr. Isaac I. Kaminer 5
   Department of Aeronautics and
   Astronautics, Code AA/KA
   Naval Postgraduate School
   Monterey, CA 93943-5101

4. Dr. Russell W. Durren 1
   Department of Aeronautics and
   Astronautics, Code AA/Ho
   Naval Postgraduate School
   Monterey, CA 93943-5101

5. Dr. Eric N. Hallberg 5
   10 Gillespie Ln
   Monterey, CA 93940

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

DUDLEY KNOX LIBRARY
3 2768 00341090 3